Oracle® Retail EFTLink

Framework Installation and Configuration Guide

Release 20.0.1

F43333-01

July 2021

ORACLE®

Oracle® Retail EFTLink Framework Installation and Configuration Guide, Release 20.0.1

F43333-01

Primary Author: Tracy Gunston

Contributor: Matthew Preston, Ian Williams

Applications. For purposes of this section, "alteration" refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

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

Content

Send Us Your Comments

Oracle Retail EFTLink Framework Installation and Configuration Guide, Release 20.0.1

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

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

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

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

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the Online Documentation available on the Oracle Help Center (OHC) website(docs.oracle.com). 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

The *Oracle Retail EFTLink Framework Installation and Configuration Guide* describes the requirements and procedures to install this Oracle Retail EFTLink release.

Audience

This Installation Guide is for the following audiences:

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

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information, see the following documents in the Oracle Retail EFTLink Release 20.0.1 documentation set:

- *Oracle Retail EFTLink Release Notes*
- *Oracle Retail EFTLink Core Configuration Guide*
- *Oracle Retail EFTLink Xstore Compatibility 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, 20.0) or a later patch release (for example, 20.0.x). If you are installing the base release, additional patch, and bundled hot fix releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch and bundled hot fix 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 (docs.oracle.com.), 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.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Overview

This chapter provides an Installation Guide Overview, a Product Overview and an Architectural Overview.

Installation Guide Overview

Installation of EFTLink consists of the following steps:

1. Extract the EFTLink files from a zip - `eftlink_v20.0 zip` to a folder on your system.
2. Select one specific core to connect to the EFT system or terminal to be used. Separate batch and script files are provided to do this for each core from a command line for both Windows and Linux.
3. Install EFTLink as a service – a batch file is provided for Windows. For Linux either the EFTLink application can be called at startup or set up as a daemon.
4. Configure the specific core.

The *Oracle Retail EFTLink Framework Installation Guide* covers the installation and configuration of the framework for EFTLink. A companion volume, the *Oracle Retail EFTLink Core Configuration Guide*, details the specific settings required to configure each Core to communicate with a specific payment system.

Product Overview

There are multiple manufacturers of Point of Sale (POS) terminals on the market. There are also large numbers of manufacturers of card readers and PIN Entry Devices (PEDs). These card readers can accept a wide variety of cards including debit cards, credit cards, loyalty cards and fuel cards for motor vehicles. These cards are provided by a wide range of issuing organizations each with their own Electronic Payment Systems (EPS). Interconnecting the POS systems, card readers and EPSs is a complex task.

EFTLink is an efficient, platform independent way of providing the connection. It is written in Java, distributed as a Java library and readily added to the software of individual POS terminals.

EFTLink is a router and protocol converter that presents a standard interface to a payment client (typically for a POS) and also links to any card readers or authorization systems in use at the retailer. The interface with the authorization system is therefore separate from the POS, removing any impact of country-specific or server-specific requirements from the POS itself.

EFTLink comes in two parts:

- The EFTLink Framework
- EFTLink Cores

The EFTLink Framework provides a system-independent execution environment (a framework) for a targeted EFT solution. The EFTLink Core for a specific terminal or payment system is implemented as a plug-in module that runs within that framework.

Oracle can provide cores for many of the most commonly used card readers or PEDs. Cores can also readily be written for any other card readers or PEDs that require them. Once a core is available for a specific device it will normally work on a range of POSs without further modification.

The POS/EFTLink interface conforms to the Open Payment Initiative (OPI). This is an open standard, widely used in the retail industry. Over time, the original OPI specification has been adopted, extended and maintained by the International Forecourt Standards Forum (IFSF). This enhanced IFSF POS-EPS version is now taken as the definitive specification.

EFTLink is not a full implementation of the IFSF POS-EPS specification. Instead, it uses those parts of the base specification that are pertinent to the sales of dry goods in the retail sector and to the sale of wet goods in petrol (gas) stations. EFTLink includes all the main messages from the IFSF POS-EPS specification and those messages contain all mandatory elements and attributes. EFTLink also includes optional elements and attributes that are commonly used by retailers.

EFTLink can also be extended beyond the IFSF POS-EPS specification. This allows additional features to be included to deal with extended payment or loyalty requirements being driven by new initiatives in retail. This gives considerable flexibility in dealing with the evolving requirements of the future.

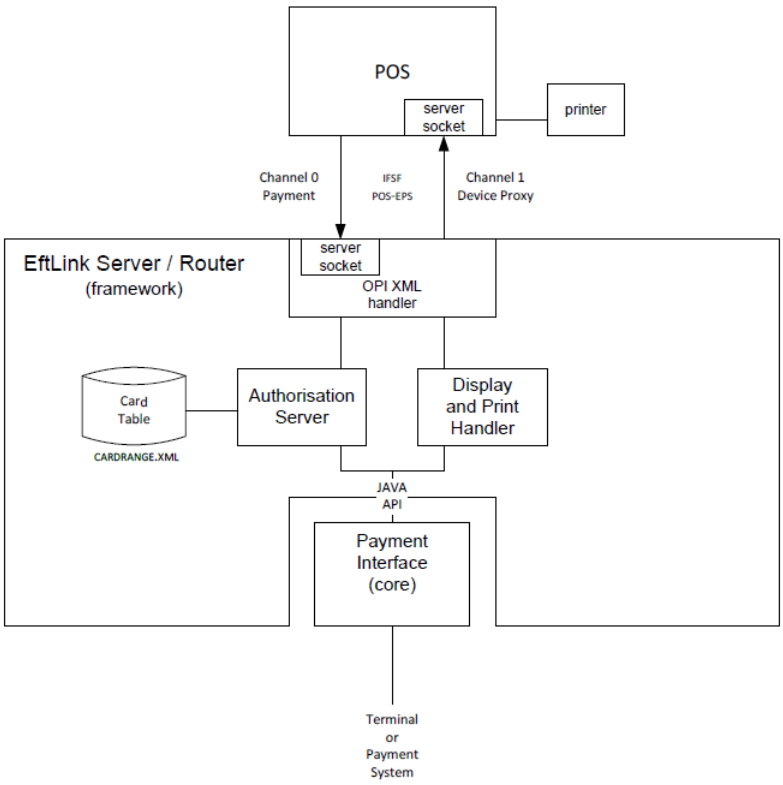
Examples of where EFTLink is used include:

- Payment, Refund, Reversal, Pre-authorization and Completion.
- Loyalty Award and Redemption, Balance inquiry, Discount voucher/coupon, IOUs.
- Stored Value Cards – Load, Redeem, Balance inquiry, Activate and so on.
- Online Agents – E-top-up and utilities payments.
- Tokenization, Gratuity, Cashback, DCC, Ad-hoc card read.
- Combined Payment and POS receipts.
- Maintenance functions.
- EPS/PED pooling.

Architectural Overview

EFTLink is a router and protocol converter, presenting an IFSF/OPI interface to a payment client (typically a POS), and linking to whatever authorization system (or systems) the customer uses. The adoption of a standard IFSF/OPI interface makes EFTLink portable to other POS or payment environments. EFTLink is not in itself a complete solution. What it provides is a system-independent execution environment (a framework) for a targeted EFT solution. The core implementation for a specific terminal or payment system is implemented as a plug-in module that runs within that framework.

Figure 1–1 Oracle EFTLink OPI Server/Router



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 down stream 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

Installation

This chapter describes the installation of EFTLink and covers the following topics:

Skillset Required

To install EFTLink successfully system implementers must:

- Understand the requirements of the specific EFT system being used, and the POS software that will be connecting to EFTLink.
- Understand the configuration settings held in property files which control how EFTLink and the selected core behave. System implementers must know how to add or modify properties within property files with their chosen text editor.
  - Java properties are case sensitive, and never contain spaces in the property name. They usually do not contain spaces in the property value – there are sometimes exceptions in lists.
  - A space is allowed before and after the = that separates the property from its value.
  - Case sensitivity does not apply to Boolean values – True is the same as true.
  - Each property = value is a separate line.
  - Lines prefixed with # are comments.

Prerequisites

EFTLink can be installed on Windows or Linux operating systems, but the procedure will differ accordingly.

Note: Oracle Retail assumes that the retailer has ensured its Operating System has been patched with all applicable Windows updates.

POS System Requirements

The POS system should meet the following minimum requirements.

- 256MB RAM
- Intel Celeron 1GHz or equivalent CPU
- 1GB disk space.

Supported Operating Systems

EFTLink is supported on the following Operating Systems:

- Oracle Enterprise Linux 7
- Windows POSReady 7
- Windows 7
- Windows 10
- Windows 10 IOT Enterprise LTSB 2016 (1607)

Java

EFTLink framework will run with any version of Java from 1. 8 whereas all strategic cores are binary compatible to Java 1.7.

EFTLink by default expects Java jre to exist in the folder location C:\jre (*on Windows*) or /opt/jre (*on a Linux kernel*).

To change the default location of java you will need to update either `include-eftlink-windows.conf` or *include-eftlink-linux.conf* which are located in `<installation directory>\wrapper\conf`.

This may be required in situations where a specific version of jre is required, such as where a different version of the jre is required to that which is being used by the POS, which may also be using the location c:\jre. See the *Oracle Retail EFTLink Core Configuration Guide* for any core jre requirements.

Installing EFTLink

Runnable Installer/Upgrader Jar

Note: This section describes how to install EFTLink using the installer jar.

Follow the steps below to install EFTLink.

The eftlink-20.x-installer.jar and eftlink-20.x-upgrader.jar are runnable and if executed will perform a silent installation/upgrade by default.

To perform a silent installation requires a pre-populated ant.install.properties file to exist within the same directory as the runnable jars.

Property Settings

Lists each mandatory setting for the ant.install.properties file.

Table 2–1 Mandatory Installer Settings

Setting	Description	Default	Example
installDir	Installs EFTLink to the directory specified. Note: When upgrading EFTLink the installDir property setting must point to the existing directory where EFTLink is installed.	C\:\\eftlink	C\:\\eftlink
eftlinkChannelZero PortNumber	Configures EFTLink eftlinkConfig.properties ServerChannel0 property setting. Note: Not applicable when running the eftlink-20.x-upgrader.jar.	10100	10100
eftlinkChannelOne PortNumber	Configures EFTLink eftlinkConfig.properties ServerChannel1 property setting. Note: Not applicable when running the eftlink-20.x-upgrader.jar.	10101	10101
selectedCore	EFTLink will install and automatically configure itself to use the class path entered here. Note: Not applicable when running the eftlink-20.x-upgrader.jar		manito.eft.tender retail.TenderReta ilCore

Performing an Install / Upgrade

1. Unzip the vxx.x.x.xxx.installer.zip file somewhere other than the desired target directory which is typically C:\eftlink or /opt/eftlink for Linux.
2. Make sure that Java is on the path of the system. In Linux, JAVA_HOME is also required to be set.
3. Navigate to the path where you extracted the installer zip file.

   For example, C:\<user>\Downloads or ~/Downloads).
4. Review the supplied ant.install.properties file and make changes if necessary. For example if performing an upgrade then ensure the installDir property setting points to the existing directory where EFTLink is currently installed.
5. Open a terminal (using elevated privilege) ensuring the directory is set to where the install/upgrader jars are located.

   Running the installer:

   a. Command to launch the installer.

      *(Windows) java -jar eftlink-(xx.x.x.x)-installer.jar or

(Linux) `sudo . java -jar eftlink-(xx.x.x.x)-installer.jar`

* Please ensure the property key "selectedCore" is populated with the desired core clashpath within the `ant.install.properties` file before running the jar. If preferred the installer jar has a graphical user interface which can be accessed during installation by adding "gui" to the end of the command statement (separated by a space). For example `eftlink-(xx.x.x.x)-installer.jar gui.`

b. The installation will end with the OPI Service being installed.

c. Within the EFTLink installation directory, copy from `C:\<eftlink installation folder>\keys` folder the pos.private.jks and eftlink.public.jks files to the POS (for example `C:\xstore\keys`).

Running the upgrader:

a. Command to launch the upgrader.

*(Windows) `java -jar eftlink-(xx.x.x.x)-upgrader.jar` or

(Linux) `sudo . java -jar eftlink-(xx.x.x.x)-upgrader.jar`

b. Once the upgrade is complete your eftlink installation directory should be updated but all configuration properties settings should have been retained.

6. Close down the terminal and remove installations files / backup files if necessary.

7. Start EFTLink. In the terminal, navigate to the installation directory, for example, `C:\eftlink` or `/opt/eftlink`.

*Windows: `start eftlink.bat`

Linux: `./eftlink.sh start`

*In Windows, you can also start the **OPI Server** in the services panel.

Manually

This section describes the installation sequence of EFTLink.

- Step 1 - Creating the EFTLink Folder
- Step 2 - Install the Files
- Step 3 - Select a Core
- Step 4 - Installing as a Service
- Step 5 - Securing Communication by Creating TLS Communication Keys
- Step 6 - Configuring the Core

Step 1 - Creating the EFTLink Folder

A folder should be created or designated for the EFTLink package. This folder can be any name and location, the only restriction is that there should be no spaces in the path. Conventionally you may wish to use the name `eftlink`.

Step 2 - Install the Files

EFTLink is supplied as a zip file, `eftlink_v20.0.zip`, and should be unzipped into the designated folder. All files needed, including the entire set of core files are included.

Once unzipped, the following files and folders should be present in the designated EFTLink folder:

Table 2–2 List of Unzipped Files and Folders

Files/Folder	Comment
apidocs	Folder containing the API documentation for the framework.
linux	Folder containing files for tanuki wrapper.
linux_64	Folder containing files for tanuki wrapper.
windows	
windows 64	
wrapper	
cores	Each core sub-directory contains the core jar file, and reference copies of that core's property file(s).
lib	The lib folder contains supporting files for EFTLink.
log	Folder containing the log files.
tmp	Working folder for EFTLink.
CardRange.xml	The default tender mapping and card identification file.
CreateKeys.bat	A batch file used to create encryption keys to ensure secure communications between POS and EFTLink.
CreateKeys.sh	A Linux script used to create encryption keys to ensure secure communications between POS and EFTLink.
eftlink.bat	A batch file used to launch the eftlink application.
eftlink.sh	A Linux script used to launch the eftlink application.
eftlink.jar	The main executable code of the EFTLink framework.
EftLinkConfig.properties	Carries the settings for the framework.
EftlinkConfig_PED_ Pool.properties	Carries the framework settings for use with PED pooling mode.
EftlinkConfig_Static_ Server.properties	
EftlinkXstore_ Mobile.properties	
Eftlink-rest-api.bat	A batch file used to launch the rest API application.
Eftlink-rest-api.jar	Executable code of the rest API application.
Eftlink-rest-api.properties	
Eftlink-rest-api.sh	A Linux shell script used to launch the rest API application.
Eftlink-rest-api-log4j2.xml	Log4j2 configuration file.
installcore.bat	A windows batch file script which sets one of cores (contained within the cores folder) as active.
installcore.sh	A Linux shell script which sets one of cores (contained within the cores folder) as active.
Jetty.xml	
LangCN.properties	Language files.
LangDE.properties	
LangEN.properties	
LangES.properties	

Table 2–2 (Cont.) List of Unzipped Files and Folders

Files/Folder	Comment
LangFR.properties	
LangIT.properties	
LangJP.properties	
LangNL.properties	
LangPT.properties	
LangRU.properties	
LangSV.properties	
Log4j2.xml	Log4j2 configuration file.

Step 3 - Select a Core

To set an active core open a terminal and change the directory to the EFTLink installation path and type:

- For Windows, `installcore.bat <core name>`
- For Linux run `installcore.sh <core name>`

For example, `installcore pointus` would set the PointUS core as the active core.

> **Note:** The core name is not case sensitive in the batch file or Linux script.

The batch or script file does two things:

- Configures EftlinkConfig.properties:

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

- Copies the selected core property file from the specific core folder to the main EFTLink folder, where it will be the active file, in this instance `pointus.properties`.

  If this is done manually you would need to edit `EftLinkConfig.properties`.

  ```
  EPSCore0=
  ```

  The value is the full classpath to the selected core application. These are the valid classpaths:

Table 2–3 Core Classpath

Core	Classpath
Adyen	manito.eft.adyen.AdyenCore
AJB FIPay	manito.eft.ajb.FIPayCore
Cayan	manito.eft.cayan.CayanCore
Merchant Link	manito.eft.poslynx.PoslynxCore
OPI Retail	oracle.eftlink.opiretail.OPIRetailCore
PayPal	oracle.eftlink.paypal.PayPalCore
Six Payment Services MPD	manito.eft.sixpay.SixpayMPDOPIClient
Tender Retail	manito.eft.tenderretail.TenderRetailCore

Table 2–3 (Cont.) Core Classpath

Core	Classpath
The Logic Group SolveConnect	manito.eft.solveconnect.SolveConnectCore
Verifone Ocius Sentinel	manito.eft.ocius_sentinel.OciusSentinelCore
Verifone Point US	manito.eft.pointus.PointUSCore
WorldPay	manito.eft.worldpay.WorldPayCore

Step 4 - Installing as a Service

This section describes how to install EFTLink as a service.

Windows Configuration

It is possible to install EFTLink as a windows service, using a third party wrapper. EFTLink is distributed with a version of Tanuki Software Limited Java Service Wrapper.

Follow the steps below on how to configure EFTLink to run as a Windows service.

1. Download and install Java.

   Ensure you have the correct version of Java installed.

   For example: if the target machine has a 64 bit OS with default 64 bit Java active but you want to use a 32 bit service wrapper, then ensure you also have the required 32 bit Java installed.

2. Installing the Service.

   a. From a command line (with administrative privileges) change to the root directory for EFTLink. For example, type `cd /eftlink`.

   b. If not already done, run installcore.bat to install the desired core which also creates and copies the necessary wrapper to .\bin. For example, type `installcore.bat adyen`.

   c. To install EFTLink as a window service, type `eftlink install`.

      If there are problems during install, it is possible to remove the service by typing `eftlink remove`. This may be necessary if the service is previously installed in a different folder. The service may then be reinstalled at the correct location by entering `eftlink install`.

   d. Once installed the service can be started and stopped from a command line:

      eftlink start

      eftlink stop

      The service can also be controlled from the Windows Services Control Panel applet ("OPI Server").

3. Examine the log file “Wrapper.log”.

   a. The log file can be found in the designated EFTLink folder\log\eftlink_ wrapper.log

   b. Installing, starting the service, stopping the service, and uninstalling the service are all briefly logged in wrapper.log, and this can be used to diagnose any problems.

Linux

It is possible to run EFTLink as a service, using a third party wrapper. EFTLink is distributed with a version of Tanuki Software Limited Java Service Wrapper.

Note: You may be required to give script file(s) execution rights. This can be accomplish by opening a terminal window and typing:

```
sudo chmod +x <PathToFile>
```

For example, `sudo chmod +x /opt/eftlink/installcore.sh`

Follow the steps below on how to configure EFTLink to run as a service.

1. Download and install Java:

   Ensure you have the correct version of Java installed.

   For example: if the target machine has a 64 bit OS with default 64 bit Java active but you want to use a 32 bit service wrapper, then ensure you also have the required 32 bit Java installed.

2. Running EFTLink.

   a. From a terminal change to the directory for EFTLink.

      For example, type `cd /opt/eftlink`.

   b. If not already done, run installcore.sh to install the desired core which also creates and copies the necessary wrapper to ./bin.

      For example, type `sudo./installcore.sh/adyen`.

   c. To run EFTLink as a service from a terminal type the following command `sudo./eftlink.sh start`.

   d. To stop, check the status or to restart EFTLink from a terminal, type one of the following commands:

      ```
      sudo./eftlink.sh stop

      sudo./eftlink.sh status

      sudo./eftlink.sh restart

      sudo./eftlink.sh condrestart
      ```

3. Examine the log file “Wrapper.log”.

   a. The log file can be found in the designated EFTLink folder\log\eftlink_ wrapper.log

   b. Starting the service and stopping the service are all briefly logged in wrapper.log, and this can be used to diagnose any problems.

Step 5 - Securing Communication by Creating TLS Communication Keys

SelfSigned Certificates

The EFTLink application does not include default TLS encryption keys for secure communication between POS client and EFTLink server, so these need to be generated as part of the installation procedure. A batch file, CreateKeys.bat, and a Linux script, CreateKeys.sh is included in the EFTLink project to facilitate creation of encryption keys.

1. Locate the `CreateKeys.bat / CreateKeys.sh` file in the EFTLink folder
2. From a terminal, run the CreateKeys script file with an appropriate set of parameters to create encryption keys.

   ```
   CreateKeys.bat-e <algorithm> <bitlength> <signAlgorithm> <daysValidity>
   CreateKeys.sh-e <algorithm> <bitlength> <signAlgorithm> <daysValidity>
   ```
   For example, `CreateKeys.bat-e RSA 4096 SHA256withRSA 750`

Table 2–4 SelfSigned Certificate Parameters

Switch	Parameter	Description	Supported Value
-e	<algorithm>	Algorithm used for TLS keys encryption.	EC,DSA,RSA
	<bitlength>	Number of bits - higher values equate to a higher level of encryption.	256 (when using EC), 1024,2048 (when using DSA), 1024,2048,3072,4096,7680,8192, 15360 (when using RSA)
	<signAlgorithm>	Signature Algorithm used.	SHA256withECDSA, SHA384withECDSA, SHA512withECDSA (when using EC), SHA256withDSA (when using DSA), SHA256withRSA, SHA384withRSA, SHA512withRSA (when using RSA)
	<daysValidity>	Number of days after creation that the certificate will remain valid.	100 to 750 days

3. Once encryption keys are created, four files will be present on the system in the keys subfolder of EFTLink:

   `pos.private.jks` to be MOVED to the POS client

   `pos.public.jks` - to remain on the EFTLink Server

   `eftlink.private.jks` - to remain on the EFTLink Server

   `eftlink.public.jks` - to be MOVED to the POS client
4. The following files should be REMOVED from the EFTLink system and placed on the POS in the folder [xstore root]\keys, where xstore root is the main POS client folder, for example: `c:\xstore\keys`:

   `pos.private.jks`

   `eftlink.public.jks`
5. This will leave only the following two files on the EFTLink server in the folder [eftlink root]\keys:

   `eftlink.private.jks`

   `pos.public.jks`
6. The removal of the appropriate files from the EFTLink server is to limit the availability of TLS keys only to where they are required, and in order to reduce the possibility of the keys being obtained and used to monitor traffic between POS and EFTLink server.

These instructions are repeated by the CreateKeys script file when keys are generated.

Note: From V20 onwards, expiry of TLS certificates is enforced by default. Self-signed certificates will be valid for a maximum of 750 days.

7. Clear warnings will be placed in log files when certificates are due to expire. Expired certificates will not result in loss of communication between POS and EFTLink.

CA Certificates

Optionally, the EFTLink application TLS encryption keys for secure communication between POS client and EFTLink server may be signed by a CA. A batch file, CreateKeys.bat, and a Linux script, CreateKeys.sh is included in the EFTLink project to facilitate creation of encryption keys, generation of signing request and import of the signed certificates.

1. Locate the `CreateKeys.bat / CreateKeys.sh` file in the EFTLink folder.
2. From a terminal, run the CreateKeys script file with an appropriate set of parameters to create encryption keys. The parameters are similar to those when used to generate self-signed certificates but specify the first parameter as -s.

   ```
   CreateKeys.bat-s <algorithm> <bitlength> <signAlgorithm> <daysValidity>
   CreateKeys.sh-s <algorithm> <bitlength> <signAlgorithm> <daysValidity>
   ```

   For example, `CreateKeys.bat-s RSA 4096 SHA256withRSA 750`

Table 2–5 CA Certificate Parameters

Switch	Parameter	Description	Supported Value
-e	<algorithm>	Algorithm used for TLS keys encryption.	EC,DSA,RSA
	<bitlength>	Number of bits - higher values equate to a higher level of encryption.	256 (when using EC), 1024,2048 (when using DSA), 1024,2048,3072,4096,7680,8192, 15360 (when using RSA)
	<signAlgorithm>	Signature Algorithm used.	SHA256withECDSA, SHA384withECDSA, SHA512withECDSA (when using EC), SHA256withDSA (when using DSA), SHA256withRSA, SHA384withRSA, SHA512withRSA (when using RSA)
	<daysValidity>	Number of days after creation that the certificate will remain valid.	100 to 750 days

3. Once encryption keys are created, a sub-folder based on the current date/time is created containing the encryption keys along with signing requests:

   For example,

   Folder name: keys20200710115046

`Eftlink.private.jks` - selfsigned file

`Pos.private.jks` - selfsigned file

`Eftlink.private.csr` - certificate signing request

`Pos.private.csr` - certificate signing request

`Eftlink.private.jks` - backup of selfsigned file

`Pos.private.jks` - backup of selfsigned file

The backup files are required for the situation where a subsequent import is attempted but does not give the required results - further attempts may be made at importing the signed certificates received from the CA.

For this reason, do not remove the backup files.

File are held in this temporary folder rather than the keys folder as the signing process may take some time, and several sets of signed keys can be handled.

4. Deliver to your CA the following files:

   `Eftlink.private.csr`

   `Pos.private.csr`

   In reply you should receive the following files (filenames may vary):

   `Eftlink.private.cer.der` - signing of EFTLink.private.csr

   `Pos.private.cer.der` - signing of POS.private.csr

   `Root.cer` - root certificate used to sign

   `Optional Intermediate.cer` - one or more intermediate certificates

5. Import the signed certificates into the keystores, by placing the signed files and root certificate (plus optional intermediate certificates) in the temporary signing keys folder keys[date] then running the following command.

   ```
   Createkeys -I <foldername> <root cert> <eftlink signed file> <pos signed file>
   <(optional) intermediate certificate 1><(optional) intermediate certificate 2>
   ```

Table 2–6 Signed Files, Root Certificates and Intermediate Certificates

Switch	Parameter	Description	Supported
-e	<foldername>	Temporary keys Subfolder name. Do not provide the full path, just the foldername.	18 character folder name
	<root cert>	The root certificate provided by the CA	Security certificate
	<eftlink signed file>	Signed file returned by CA	Security certificate
	<pos signed file>	Signed file returned by CA	Security certificate
	<intermediate certificate 1>	CA Intermediate certificate	Optional Security certificate
	<intermediate certificate 2>	CA Intermediate certificate	Optional Security certificate

For example, `createkeys -i keys20200101010101 ca_root.cer eftlink.private.der.cer pos.private.der.cer ca_intermediate1.cer ca_intermediate2.cer`

6. Archive the temporary keys[date] folder to a safe location as this contains sensitive information.
7. The following files should be REMOVED from the Eftlink system and placed on the POS in the folder [xstore root]\keys, where xstore root is the main POS client folder, for example: `c:\xstore\keys`:

   `pos.private.jks`

   `eftlink.public.jks`
8. This will leave only the following two files on the EFTLink server in the folder [eftlink root]\keys:

   `eftlink.private.jks`

   `pos.public.jks`
9. The removal of the appropriate files from the EFTLink server is to limit the availability of TLS keys only to where they are required, and in order to reduce the possibility of the keys being obtained and used to monitor traffic between POS and EFTLink server. These instructions are repeated by the CreateKeys script file when keys are generated.

   Note: From V20 onwards, expiry of TLS certificates is enforced by default. Self-signed certificates will be valid for a maximum of 750 days.
10. Clear warnings will be placed in log files when certificates are due to expire. Expired certificates will not result in loss of communication between POS and EFTLink.

Step 6 - Configuring the Core

See the *Oracle Retail EFTLink Core Configuration Guide* located on `OHC` and refer to the chapter for the specific core selected.

Post Installation Steps

By default in Windows, the 'OPI Server' service is using the Local system account user. In order to ensure for EFTLink service to create dynamic key store files, a user with an administrative privilege is needed. This is only applicable for cores like PointUS and Cayan. In the services panel, right click on the OPI Server service. Select the **Properties** option. Select the **Log on** tab. Select **This account:**. Input the user's credentials and select **OK**.

- Adyen: The POS_JNI jar which is provided by Adyen is also required. This needs to be copied to `C:\eftlink\cores\Adyen` or `/opt/eftlink/cores/Adyen` for Linux. Refer to the **Third Party** section of the Adyen core in the *Oracle Retail EFTLink Core Configuration Guide* located on `OHC` for more details.
- AJB FiPay: The `AJBComm.jar` component needs to be copied to `C:\eftlink\cores\FIPay` or `/opt/eftlink/cores/FIPay` for Linux. Refer to the **FileSet** section of the AJB core in the *Oracle Retail EFTLink Core Configuration Guide* located on `OHC` for more details.
- Cayan: The merchant credentials which are supplied by Cayan team are needed to be setup. This can be done in Xstore's back office through the EFTLink Admin

functions. Refer to the **Account Information Entry** section of the Cayan core in the *Oracle Retail EFTLink Core Configuration Guide* located on OHC for more details.

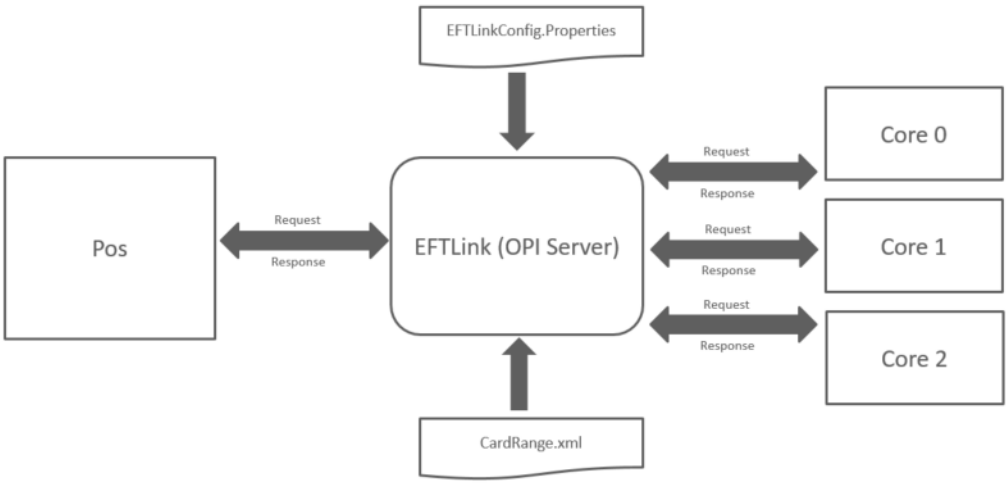
- VerifoneUS: The PED needs to be paired with EFTLink prior to use. This can be done through Xstore's back office in the EFTLink Admin functions. Refer to the **Administration Functions** section of PointUS core in the *Oracle Retail EFTLink Core Configuration Guide* located on OHC for more details.

EFTLink Multiple Core Feature

EFTLink can be configured to use multiple cores for the purpose of redirecting requests based on:

- Line display
- Request Type
- Cardrange.xml
- Referrals

Figure 2–1 Multiple Cores



Line Display

Sale State Notification requests can either be disabled altogether or be forwarded to either one particular core or a delegated list of cores.

Preselected Cores

Preselected cores are configurable within the `eftlinkconfig.properties` file. You can configure EFTLink to forward requests to a particular core based upon EWallet, GiftCard (Card Service Request) or a Custom Form (Device Request).

CardRange.xml

EFTLink always checks the cardrange.xml before determining which core is selected for handling the card Service Request.

The cardrange.xml offers EFTLink the ability to redirect card service requests to preselected cores base on the *card pan or card type.

*As the POS is not subject to sending an EMV PAN due to PCI regulations. Card PAN is only applicable with non PCI transactions (Gift card or Ewallet).

Referrals

EFTLink supports a referral feature whereby a core can request that another core handles the request on its behalf.

By default EFTLink sets the main core (core 0) for referrals however this is configurable via `EFTLinkConfig.Properties`.

The cardrange.xml can also be used to redirect the referral to any active core.

A referral could be based upon a feature not supported ***or***, it could be based upon a particular failed response / error code.

The core requesting the referral is in control and is responsible for the transaction response back to the POS.

EFTLink Server

EFTLink is usually deployed as a service application running on each POS and connecting to a single payment device. To support environments where the POS runs as a thin-client application with restricted local device access, or where the hardware has limited processing power or memory, EFTLink can be deployed in Store Server mode. A single EFTLink application runs on a designated server system and all POSs connect to that one server. EFTLink manages the connections to multiple payment terminals and routes payment requests from each POS on to the relevant device.

Generally, using Server mode, there is still a 1-1 logical connection between POS and payment terminal, but it is also possible for EFTLink to make a dynamic selection of payment terminal based on availability and convenience. This is referred to as PED-pooling (PED - PIN entry Device).

Similarly, the EFTLink Server can be used to manage a pool of printers shared between the POSs and allocated dynamically. This is referred to as Print-pooling.

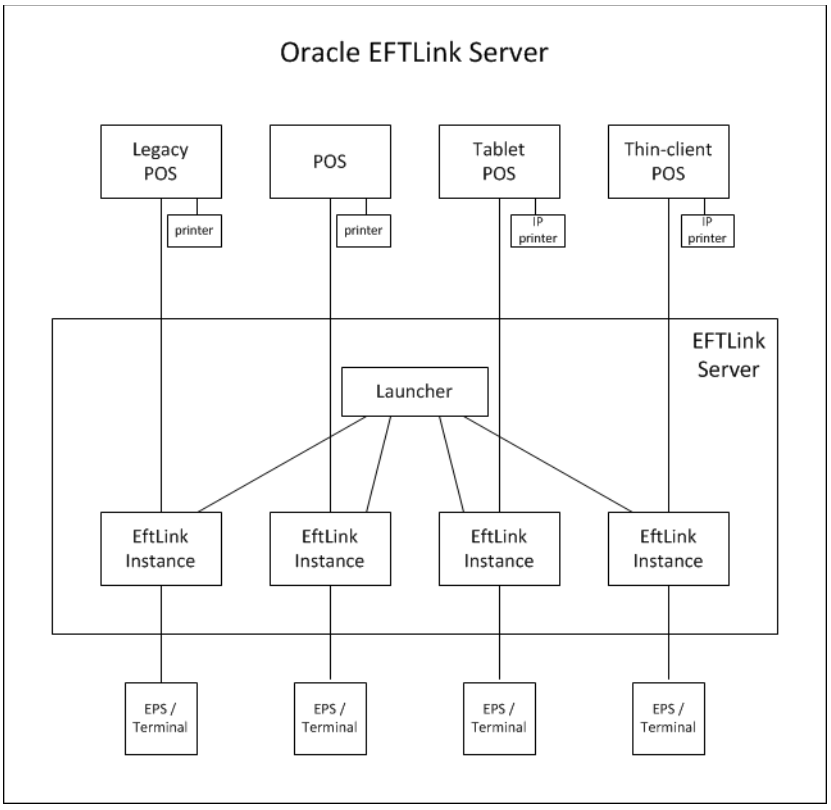
This solution is only really possible with IP-based payment terminals and printers. The server system should be in a secure room, and the terminals/printers spread around the store, so direct wired connections are not practical.

The standard `EFTLinkConfig.properties` will ensure EFTLink is configured for use as an EFTLink Server.

EFTLink Server - Remote Mode

1-1 mapping between the POS and payment system/terminal. Each POS is allocated a fixed pair of sockets (channel 0/1) that connect to a dedicated EFTLink instance.

Figure 2–2 EFTLink Server - Remote Mode



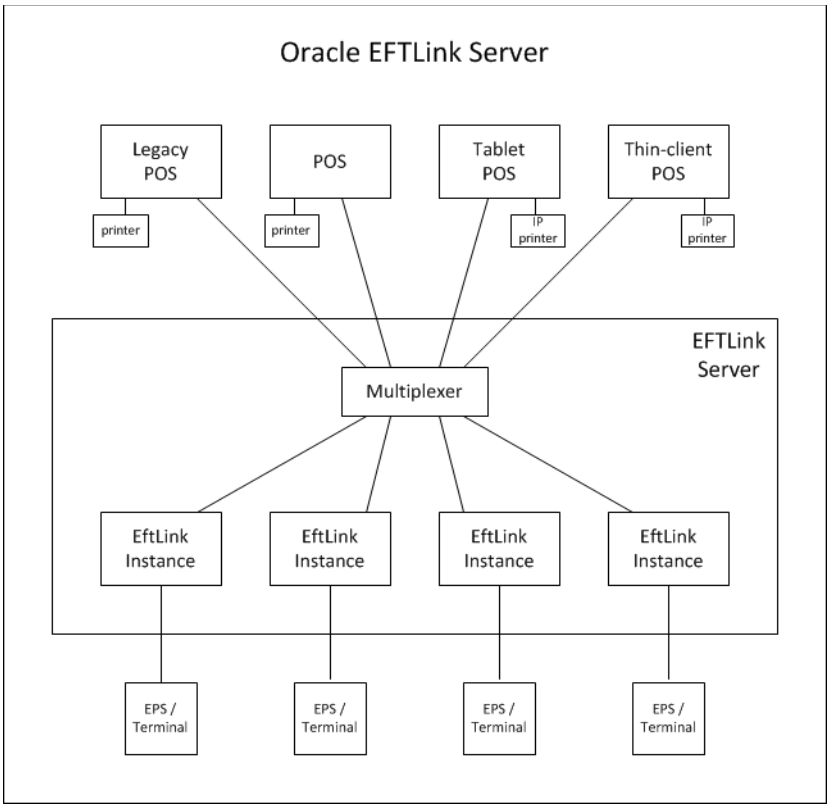
Included with EFTLink is an additional file `EFTLinkConfig_Static_Server.properties`. This is a sample file demonstrating EFTLink configuration in this mode.

`EFTLinkConfig_Static_Server.properties` can be used in place of the standard `EFTLinkConfig.properties` by renaming this file to `EFTLinkConfig.properties`. A manual comparison of the files will be necessary to ensure core configuration which is set during installation is copied over to the RemoteMode configuration.

EFTLink Server - PEDPool Remote Mode

Many-many mapping between POS and payment system/terminal. Each POS is allocated a fixed pair of sockets (channel 0/1) that connect to a multiplexer/switch. The multiplexer implements rules and/or uses interactive dialogs with the POS operator to determine which EFTLink instance to pass the request on to.

Figure 2–3 EFTLink Server - PEDPool Remote Mode



Included with EFTLink is an additional file `EFTLinkConfig_PED_Pool.properties`. This is a sample file demonstrating EFTLink configuration in this mode.

`EFTLinkConfig_PED_Pool.properties` can be used in place of the standard `EFTLinkConfig.properties` by renaming this file to `EFTLinkConfig.properties`. A manual comparison of the files will be necessary to ensure core configuration which is set during installation is copied over to the PEDPool configuration.

Configuring EFTLink Server

Configuring/deploying EFTLink Server is rather more complicated that standard EFTLink and is currently only possible as a manual procedure.

As a base, EFTLink should first be installed on the chosen server system using the standard installation procedure.

- Enabling Server Mode
- EFTLink Instance Set Up
- PED Pooling Set Up
- Xstore Set Up

Enabling Server Mode

EFTLink Server uses a different main class from normal.

When not using the standard Tanuki wrapper / eftlink.bat file to start eftlink, replace the following lines where applicable in the startup file:

Windows

Replace: `java manito.eft.opi.server.OPIServer`

With: `java manito.eft.opi.server.MultiServerLauncher`

Linux

Replace: `java -cp $CLASSPATH manito.eft.opi.server.OPIServer`

With: `java -cp $CLASSPATH manito.eft.opi.server.MultiServerLauncher`

Tanuki Wrapper Configuration

Use a text editor to edit EFTLink folder/wrapper/conf/eftlink.conf.

Replace: `wrapper.app.parameter.1=manito.eft.opi.server.OPIServer`

With: `wrapper.app.parameter.1=manito.eft.opi.server.MultiServerLauncher`

This can be sone by commenting out all wrapper.app.parameter.1 and license details for manito.eft.opi.server.OPIServer and uncomment all license details for manito.eft.opi.server.MultiServerLauncher in the section below.

PED Pool

Replace: `PEDPoolEnabled = false`

With: `PEDPoolEnabled = true`

Replace: `PEDPoolOneCatchAllChannel0 = false`

With: `PEDPoolOneCatchAllChannel0 = true`

See PED Pooling Set Up for more information.

EFTLink Instance Set Up

Each instance of EFTLink is identified by a unique sequence number starting from 1.

For each instance of EFTLink required (that is, for each payment terminal):

1. In the main eftlink folder, run installcore.bat as if configuring standalone EFTLink. This will setup the EftlinkConfig.properties file.
2. Create a subfolder under the main eftlink folder named serverN, where N is the sequence number.
3. Copy all properties files (*.properties) from the main eftlink folder into the new serverN folder.

   This excludes the sample files EftlinkConfig_PED_Pool.properties, EftlinkConfig_Static_Server.properties and EftlinkConfig_Xstore_Mobile.properties. EFTLink and core specific files are required, including language files. For some cores, additional files may also need to be copied over (such as receipt.txt files) - to see the full list of required files, refer to the cores\[corename] sub-folder.
4. Using a text editor, edit the core-specific properties file in the subfolder to set any properties that are unique for each core instance for example, the terminal IP.

5. Using a text editor edit EftlinkConfig.properties in the main eftlink folder:

   Find the NumServers setting and change it to be the number of EFTLink instances to be used. Un-comment (that is, remove the leading '#' if present) if necessary. For example, NumServers = 2.

6. For each EFTLink instance, assign a descriptive title. These are the names that will be presented to the operator and should identify the relevant payment terminal in some way such as by its location, for example:

   server1.description = Menswear-suits

   server2.description = Menswear-paydesk #2 till 1

   Note: Spaces are allowed in the descriptive names, but not commas if PED pooling is to be used.

Log4J2 Setup

The Log4j2.xml logging configuration file as standard is delivered configured for Single server mode. Alterations are required to the log4j2.xml file to ensure logging is performed per pos, and per server. To enable full logging, modify the standard log4j2file by performing the following steps:

1. Alter the <Properties> section, adding in the correct number of servers, and pos, ensuring each has a unique name and filename.
2. In the <Appenders> section, enable the RollingRandomAccessFile entries for each server/pos by removing the comment start <!-- and comment end --> for the marked MultiServerLauncher/PedPooling section.
3. Adjust the number of the RollingRandomAccessFile entries in the <Appenders> section by adding the relevant number of server{x}_log and pos{x}_log sections. Ensure each of these maps to the correct filename (defined in point 1) and also adjust the filepattern to use the relevant server folder / server filename. The number of server{x}_log and pos{x}_log entries in the <Appenders> section should match the number of server{x}_log and pos{x}_log entries in the <Properties> section.
4. Also in the <Appenders> section, enable the Async entries for each server/pos by removing the comment start <!-- and comment end --> for the marked MultiServerLauncher/PedPooling section.
5. Adjust the number of the Async entries in the <Appenders> section by adding the relevant number of server{x}_log and pos{x}_log sections. Ensure each of these maps to the correct server{x}_log or pos{x}_log (defined in point 3).
6. In the <Loggers> section, enable the Logger entries for each multifile.server{x}/multifile.pos{x} by removing the comment start <!-- and comment end --> for the marked MultiServerLauncher/PedPooling section.
7. Adjust the number of the Logger entries in the <Loggers> section by adding the relevant number of multi-file.server{x} and multifile.pos{x} sections. Ensure each of these maps to the correct async_server{x}_log or async_pos{x}_log (defined in point 5).

Once fully configured, each pos request will write to a file in the main eftlink log folder named pos{x}.log. In addition, each server folder will contain its own log file showing server processing of the request - log files for each server will be in the path server{x}/log/server{x}.log.

POS Client Set Up

Each POS client is identified by a unique sequence number starting from 1.

1. Use a text editor to edit EftlinkConfig.properties in the main eftlink folder:

   Find the NumClients setting and change it to be the number of POSs that will be using EFTLink. Un-comment (that is, remove the leading '#' if present) if necessary. For example, NumClients = 2

2. For each POS, assign a descriptive title. These are the names will be shown in the EFTLink log to ease tracking/debugging, for example:

   pos1.description = Menswear-suits

   pos2.description = Menswear-mobile#1

3. EachPOS has to use a unique pair of ports for its connection to EFTLink. These do not need to be further defined within EftlinkConfig.properties, but the ports numbers and EFTLinkServer system IP must be set on each POS. The numbering system is based on EFTLink base address (default 10100, configurable by the ServerChannel0 property) plus 10 x the POS number. Two sequential ports are needed, one for each of channel 0 and 1. This gives a default allocation of:

   POS1 - 10110/10111

   POS2 - 10120/10121

   POS3 - 10130/10131

   ...

   POS9 - 10190/10191

   POS10 - 10200/10201

   POS11 - 10210/10211

   and so on

   If this range of ports is not available, the base number can be changed via the ServerChannel0 setting. All POSs must then be changed to match.

PED Pooling Set Up

If PED pooling has been enabled, the system uses the standard channel 1 display messages to present each POS operator with a list of available payment terminals. By default, the list will include all available terminals, but this can be confusing in a large store, so there is an option to limit each POS to a subset of the full list to show just the terminals in one department. The subset is defined using the descriptive names from EFTLink Instance Set Up, and specified as a comma-separated list. A default association can be set by prefixing the descriptive name with '*'. If that payment terminal is available, it will be automatically used without any operator prompting.

For example:

pos1.subpool = *Menswear-suits

pos2.subpool = Menswear-suits, *Menswear-paydesk #2 till 1, Menswear-paydesk #2 till 2

Note: It is important to point out that the EFTLink PED pooling functionality is restricted by Core compatibility. Please note the following restrictions:

PED pooling is only applicable within the <CardServiceRequest> context, that is, this is when the actual payment is initiated and finalized.

PED pooling is not currently applicable within the <SaleStateNotification> context, that is, if the EPS supports a device that is dependent on a line display, this functionality will need to be suppressed by Xstore or the Core (depending on configuration).

PED pooling is not possible where the EPS requires the register to be paired with a single device thereby forcing a one to one relationship between the register and the device.

Xstore Set Up

As noted above, each POS has to use a unique pair of ports for its connection to EFTLink. Also, the POS is configured to access a remote EFTLink rather than a local one.

There are two different ways that Xstore can be set up to use with EFTLink in Server Mode.

- One to One Port Mapping (applies to both Xstore and Xstore Mobile)
- One to Many Port Mapping (applies to both Xstore and Xstore Mobile)

All configurations illustrated below are part of the Xstore AuthConfig.xml configuration file.

One to One Port Mapping (Static Server Mode)

This is where there is one Xstore or Xstore Mobile client served from the Jetty instance. It will divert all requests to a single port pairing that is managed inside the EFTLink Server instance. If another POS client is configured to use the same port pairing, it will potentially be blocked out until the port pair becomes free. In this mode, EFTLink Server will allow a single device to use many PEDs through the PED pooling functionality. EFTLink Server does not support load balancing of requests through one port pair so this configuration is not recommended if there are many Xstore mobile clients in the store solution.

If this configuration is suitable then the Xstore Mobile configuration is identical to the standard Xstore configuration. The 'communicatorHosts' parameter is used to set the channel 0 URL and 'deviceCommChannel' is used to set the channel 1 URL, as illustrated below. In this configuration when Xstore or Xstore Mobile starts an authorization request EFTLink will process the authorization request in the expected way, or if PED pooling is enabled, it will send a list of available PEDs for an associate to choose. Once the associate has chosen a PED, the authorization will proceed in the expected way.

```
<AuthProcess name="EFT_LINK_HOST" Abstract="true">
    <Parameter name="communicatorHosts">
      <param_value dtype="List">
        <Host dtype="String">socket://localhost:10100;timeout=1000</Host>
      </param_value>
    </Parameter>
    <Parameter name="deviceCommChannel" value="socket://localhost:10101" />
```

```
...
...
    <Parameter name="additionalWorkstationHostsMap">
      <param_value dtype="Map">
        <MapEntry>
          <key dtype="Integer">1</key> <!-- workstation id -->
          <value dtype="EFTLinkCommunicationChannels">
            <Channel0 dtype="String">socket://localhost:10110</Channel0>
            <Channel1 dtype="String">socket://localhost:10111</Channel1>
          </value>
        </MapEntry>
       <MapEntry>
          <key dtype="Integer">2</key> <!-- workstation id -->
          <value dtype="EFTLinkCommunicationChannels">
            <Channel0 dtype="String">socket://localhost:10120</Channel0>
            <Channel1 dtype="String">socket://localhost:10121</Channel1>
          </value>
        </MapEntry>
      </param_value>
    </Parameter>
  </AuthProcess>
```

One to Many Port Mapping (PED Pooling)

In order to setup Xstore this way, the EftlinkConfig.properties in the main folder in EFTLink (for example, C:\eftlink) should be copied in the working directory of Xstore or Xstore mobile (for example, C:\xstore or C:\xstoremobile). The list of POS should be the same as in the EFTLink server side.

pos1.description = POS 1

pos2.description = POS 2

pos3.description = POS 3

The additional WorkstationHostsMap parameter is not needed anymore. If the default channel zero is used (for example, ServerChannel0 = 10100), then make sure to update the port in the Host section of the communicatorHosts to 10110. If ServerChannel0 is different, simply add 10 to it. Then deviceCommChannel's port is plus 1 of the Host's port.

```
<AuthProcess name="EFT_LINK_HOST" Abstract="true">
    <Parameter name="communicatorHosts">
      <param_value dtype="List">
        <Host dtype="String">socket://localhost:10110;timeout=1000</Host>
      </param_value>
    </Parameter>
    <Parameter name="deviceCommChannel" value="socket://localhost:10111" />
...
...
  </AuthProcess>
```

Included with EFTLink is an additional file `EFTLinkConfig_XStore_Mobile.properties`. This is a sample file demonstrating the required settings for the file `EFTLinkConfig.properties` on the POS.

This file should be copied over the POS Client as `EFTLinkConfig.properties`.

3

EFTLink Configurable Properties

This chapter describes the EFTLink properties:

Configuration Settings

The full set of configuration properties are defined and commented in `EftlinkConfig.properties`.

Key Settings

These settings must be set for all POS.

Table 3–1 Key Settings

Setting	Description	Example
EPSCore0	Name of EPS subsystem. Plugin cores must be specified by their full package name, and the package must also be added to the execution class path. EPSCore0 is mandatory. Note: EPSCore0 is set by `installcore.bat / installcore.sh`.	`EPSCore0 = manito.eft.pointus.PointUSCore`
DisplayLanguage	Language for display texts. For whichever country code is set, there must be a matching `LangXX.properties` file. A hierarchy is implied for example EN_US is taken as an extension of EN.	`DisplayLanguage = EN`
LanguageFolder	The location of the `Lang<CC>_<Core>.properties` files exist. Support relative path. Not permitted to traverse outside of installation folder.	`./lang`

Secondary Settings

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

Table 3–2 Secondary Settings

Setting	Description	Default	Example
NumEPSCores	The number of active EPS cores list specified by EPSCore<n>	1	`NumEPSCores = 2`
ServerChannel0	Socket that EFTlink listens on for incoming Channel 0 requests from POS.	10100	`ServerChannel0 = 10100`
ServerChannel1	Socket that EFTlink uses to send Channel 1 Device Requests to POS.	10101	`ServerChannel1 = 10101`
Channel1IP	IP that EFTlink uses to send Channel 1 Device Requests to POS.	localhost	`Channel1IP = IP ADDRESS`
TLSEnabled	Whether to use Transport Layer Security (TLS) between the core and the framework.	true	`TLSEnabled = true`
TLSExpiry	Specify whether to enforce expiry of TLS certificates, based on expiry date. Note. Self-certified certificates created by the "CreateKeys" script files will expire after a maximum of 750 days.	true	`TLSExpiry = false`
TLSExpiryWarningLogDays	Specify the number of days prior to TLS certificate expiry that clear warnings will be included in log files during communication sessions.	90	`TLSExpiryWarningLogDays = 180`
TLSExpiryWarningMessageDays	Specify the number of days that clear warnings presented to the operator at start of day prior to TLS certificate expiry.	90	`TLSExpiryWarningMessageDays = 90`
OPIServerDelegate	Allows the OPIServer operation to be delegated to an alternate class		`OPIServerDelegate = manito.eft.tlog.TLogOPIServer`

Table 3–2 (Cont.) Secondary Settings

Setting	Description	Default	Example
InvalidCorePromptTimeout	Timeout in seconds for displaying the TXT_INVALID_CORE message to the operator.	10	`InvalidCorePromptTimeout = 5`
SingleSocket	Whether EFTLink is to be accessed via a single common server socket, with messages routed by POS ID Note: In this mode, channel 1 will run on the same client socket as channel 0.	false	`SingleSocket = true`
LineDisplayEnabled	If set to false all Sale State Notifications will be ignored and not passed on to any active EPSCore.	true	`LineDisplayEnabled = false`
DelegateLineDisplay	If set to true a delegated list will be used to control which core receives Sale State Notification requests. Applicable only when 'DelegateLineDisplay' is set to true.	false	`DelegateLineDisplay = true`
LineDisplayDelegateList	A comma separated list of all cores that are to receive Sale State Notification requests.		`LineDisplayDelegateList = 0,1,2`
EwalletCore	A particular core can be designated to handle EWallet operations.	0	`EwalletCore = 1`
GiftCardCore	A particular core can be designated to handle Gift Card operations.	0	`GiftCardCore = 1`
CustomFormCore	A particular core can be designated to handle custom forms operations.	0	`CustomFormCore = 1`
ReferralCore	A particular core can be designated to handle Referrals.	0	`ReferralCore = 1`
SelfReferralEnabled	Whether to allow a core to handle its own referral.	false	`SelfReferralEnabled = true`

Table 3–2 (Cont.) Secondary Settings

Setting	Description	Default	Example
NumServers	Determines how many instances of the OPIServer to enable in server mode. In normal stand alone or non-server mode, set this to 0.	0	`NumServers = 1`
PEDPoolEnabled	Whether to enable PED pooling in server mode. The NumServers should be set to a number greater than zero. In PED pooling mode, the PEDs can be shared among POS clients.	false	`PEDPoolEnabled = true`
PEDPoolOneCatchAllChannel0	Whether to open just one port for channel zero in PED pooling mode.	false	`PEDPoolOneCatchAllChannel0 = true`
Server<n>.description	The list of server or PED identifier. This is mandatory when in PED pooling. *n is a positive number starting at 1 and up to NumServers above.		`c`
NumClients	Determines how many potential clients when using PED pooling. This is mandatory in PED pooling.	2	`NumClients = 1`
posN.description	The list of POS identifier where N is a positive number starting at 1. This is mandatory in PED pooling.		`pos1.description = POS 1`

posN.subpool	Restrict the list of server or PED for a particular POS where N is the workstation ID. A default association can also be specified by prefixing the server ID with '*'.	null	`pos1.subpool = *EFT 1, EFT 2`
			`pos2.subpool = EFT 1, EFT 2`
			In the above example, register 1 by default will use EFT 1 if it's free. Both EFT 1 and EFT 2 servers is available for both registers (1 and 2).

Table 3–2 (Cont.) Secondary Settings

Setting	Description	Default	Example
ProtocolsWhiteList	Restricts the protocols which are permissible in the connection between POS and EFTLink Server. Default only allows for TLS 1.2 security.	SSLv2Hello,TLSv1.2	`ProtocolsWhiteList= SSLv2Hello, TLSv1.2`

CipherWhiteList	Restricts the ciphers which are permissible in the connection between POS and EFTLink Server. The whitelist only includes ciphers which are approved under Oracle Approved Technologies: Security Protocols.	TLS_DHE_.*_WITH_ AES_128_.*,
		TLS_ECDHE_.*_ WITH_AES_128_.*,
		TLS_ECDH_.*_ WITH_AES_128_.*,
		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_ECDHE_ ECDSA_WITH_AES_ 128_CBC_SHA256,
		TLS_ECDHE_RSA_ WITH_AES_128_ CBC_SHA256,
		TLS_RSA_WITH_ AES_128_CBC_SHA
			`CipherWhiteList =`
			`TLS_DHE_.*_WITH_ AES_128_.*,`
			`TLS_ECDHE_.*_WITH_ AES_128_.*, TLS_ ECDH_.*_WITH_AES_ 128_.*,`
			`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_ECDHE_ECDSA_ WITH_AES_128_ CBC_ SHA256,`
			`TLS_ECDHE_RSA_WITH_ AES_128_ CBC_ SHA256,`
			`TLS_RSA_WITH_AES_ 128_CBC_SHA`

CipherBlackList	CipherBlackList	SSL_.*,
		TLS_EMPTY_.*,
		.*_SHA,
		.*_3DES_.*,
		.*_DES_.*,
		.*_WITH_NULL_.*,
		.*_anon_.*,
		.*EXPORT.*,
		.*LOW.*,
		.*MD5.*,
		.*DES.*,
		.*RC2.*,
		.*RC4.*,
		.*PSK.*
			`CipherBlackList=`
			`SSL_.*,`
			`TLS_EMPTY_.*,`
			`.*_SHA,`
			`.*_3DES_.*,`
			`.*_DES_.*,`
			`.*_WITH_NULL_.*,`
			`.*_anon_.*,`
			`.*EXPORT.*,`
			`.*LOW.*,`
			`.*MD5.*,`
			`.*DES.*,`
			`.*RC2.*,`
			`.*RC4.*,`
			`.*PSK.*`

Table 3–2 (Cont.) Secondary Settings

Setting	Description	Default	Example
PosType	POS type that EftLink is connected to. This can be set explicitly (e.g. Lucas, Retail-J, Oscar) or set to "Auto" for the POS type to be deduced from the OPI	Auto	`PosType = Auto`
Dynamic Configuration	Static/Dynamic Configuration EFTLink can be configured to pick up its configuration dynamically from POS messages. A default setting is implied by the POS type setting, but this can be overridden.	false	`DynamicConfiguration = false`
PosIfsfCompliance	The level of IFSF compliance for the POS interface - IFSF or LUCAS.	Lucas	`PosIfsfCompliance = Lucas`
Decimal Places	Number of decimal places to show.	2	`DecimalPlaces = 2`
DelegatedDisplay	Whether to use a display server delegate class to control pop-up dialogs directly from EFTLink instead of via Channel1.	false	`DelegatedDisplay = true`
DelegatedDisplayHandler	Class implementing pop-up dialogs.	manito.deviceproxy.DeviceProxy	`DelegatedDisplayHandler = manito.deviceproxy.DeviceProxy`
DelegatedDisplayOverride	Optional override to revert some display operations back to the POS.	0	`DelegatedDisplayOverride = 0`
ShowPrintingDialog	Whether to precede each print request with a TXT_ PRINTING (for example, "Printing. Please Wait") dialog.	false	`ShowPrintingDialog = false`
ForcedInput	Whether to request forced input (no cancellation) on input requests to the POS, if not explicitly set by the core.	false	`ForcedInput = true`

Table 3–2 (Cont.) Secondary Settings

Setting	Description	Default	Example
DeviceEvents	Whether device events such as CardInserted are supported by the POS. Default false.	false	`DeviceEvents = false`
PrinterPoolEnabled	Whether to run a pool of printers shared between POSs. (many-many link) Printer pool is accessed via the "master" channel 0. Channel 1 will run on the same client socket as channel 0.	false	`PrinterPoolEnabled = true`
PaymentWithLoyalty	Whether combined payment with loyalty is supported. Combined payment with loyalty is automatically disabled if a part payment is detected.	true	`PaymentWithLoyalty = false`
ValidateItemValues	Whether the basket content should be validated to ensure that the sum of the items matches the overall value. Default true.	true	`ValidateItemValues = true`
PrinterImpliedOnline	Whether the printer can be assumed to be online and available, that is, if the POS can only send requests when the printer is online and with paper, there is no need to do an explicit check.	false	`PrinterImpliedOnline = false`
ClearDisplayAfterTimeout	Whether to clear the display by sending an empty prompt to the POS after a timeout.	false	`ClearDisplayAfterTimeout = false`

CURRENCY_< currency symbol>	Currency symbol conversion list.		`CURRENCY_156 = GBP`
			`CURRENCY_163 = GBP`
			`CURRENCY_164 = EUR`
			`CURRENCY_213 = EUR`

DespoolOnLogon	Spooled reports are automatically printed on next logon.	false	`DespoolOnLogon = true`

Table 3–2 (Cont.) Secondary Settings

Setting	Description	Default	Example
DespoolOnMaintenance	Spooled reports are automatically printed on next maintenance/administration use.	true	`DespoolOnMaintenance = false`
DespoolOnReconciliation	Spooled reports are automatically printed at next shift close.	true	`DespoolOnReconciliation = false`
DistributedDayend	Whether EFTLink is to relay POS reconciliation message on to other instances of EFTLink. # If set true, EFTLink uses the same day end client list as for manito.eft.opi.server. Dayend	false	`DistributedDayend = false`
NumDayendClients	List of client systems to which a reconciliation message should be sent by the manito.eft.opi.server. Dayend operation. Number of clients to be processed.	0	`NumDayendClients = 1`
DayendClient<n>IP	IP of remote system where EFTLink is running.		`DayendClient0IP = xxx.x.x.x`

DayendClient<n>Channel0	Port which EFTLink is running.		`DayendClient0Channel = 10100`
			`DayendClient1Channel0= 10100`
			`DayendClient2Channel0= 10100`
			`DayendClient3Channel0= 10100`
			`DayendClient4Channel0= 10100`

DayendClient<n>Batch	Batch file to be run locally instead of sending message.		`DayendClient0Batch = dayend.bat`
DayendClient<n>Core	Specific individual core to send the request to.		`DayendClient0Core = EftDevice`

Table 3–2 (Cont.) Secondary Settings

Setting	Description	Default	Example
AllowMapMachineNameToSystemAccount	Allow the application to correctly secure access to data folders when running under the Windows Local System Account. It is strongly recommended that the application is not configured to run using the Windows Local System account, instead use the Windows Local Service account when use of a local Windows machine account is desired. Note that the Windows Network Service account should not be used.	false	`AllowMapMachineNameToSystemAccount = false`
https.proxyHost	Sets the https proxy host.		`https.proxyHost=adc-proxy.example.com`
https.proxyPort	Sets the https proxy port.		`https.proxyPort=80`
http.proxyHost	Sets the http proxy host.		`http.proxyHost=adc-proxy.example.com`
http.proxyPort	Sets the http proxy port.		`http.proxyPort=80`
ImagePathWhitelist	Comma delimited list of permissible paths for image files used in device request XML. For example, c:/Images,c:/eftfolder/resources/images 'Any' or a blank can be used but having no entry serves the same purpose.		`ImagePathWhitelist = Any`
DisplayListOfPEDForFailure	In PED pooling mode, this determines if Eftlink displays the list of PEDs when the request failed or declined using the default PED. This is to give the user an option to select another PED in the next request.	false	`DisplayListOfPEDForFailure = false`

Table 3–2 (Cont.) Secondary Settings

Setting	Description	Default	Example
CardRangeFile	The name and location of the range xml file.	Defaults to cardrange.xml file located within the root of the eftlink installation directory	`./rangefile/cardrange.xml`

MultiJVM

Note: This functionality is currently incompatible with the `PEDPoolEnabled` property.

This property is used to launch each OPIServer in their own Java Virtual Machine (JVM) process when the NumServers property is set to greater than 0.

Each server's channel 0 and channel 1 ports are based on the ServerChannel0 setting. For example; if the ServerChannel0 is set to 10100 and NumServers is set to 3, the additional servers will be created on channel 0 ports 10110, 10120, 10130 and the corresponding channel 1 ports will be 10111, 10121, 10131 therefore, you must ensure that these ports are available for use with EFTLink.

For each server defined under NumServers; EFTLink looks for a corresponding server folder. For example, if NumServers is set to 3, EFTLink looks for server folders named server1, server2 and server3 under the EFTLink directory. These folders must contain their own configuration files, that is; EftLinkConfig.properties and so on.

In order to use this property you must use the MultiServerLauncher application rather than the OPIServer application.

Note: It is not recommended to use this functionality with compact systems where memory is at a premium. Therefore, taking the above example of 3 servers running in separate JVMs with each JVM taking roughly ~60MB of memory, EFTLink would require at least ~180MB of free memory (this is a purely hypothetical situation, actual memory usage may be system-dependent).

There may be additional memory requirements dependent upon the core being used with EFTLink.

You must ensure when, using this functionality, that a clean shutdown of EFTLink is performed in order to destroy the child processes which have been created. In Windows command line; the command CTRL+C is used to terminate a batch job cleanly, which will close EFTLink when running using the included batch file/Tanuki wrapper.

Example:

```
MultiJVM = false
```

4

EFTLink General Information

This chapter provides general information about EFTLink:

Tender Mapping

EFTLink provides a table - CardRange.xml - for mapping EFT cards to POS tenders. This is done by card IIN range, or, where that is not possible, by card name (also known as card circuit). The resulting numeric code is returned to the POS so that it can determine which tender to allocate the payment to. By default the table maps all card to a single "type" (or tender) by a simple wildcard catchall. This can be used as-is, but if a more detailed breakdown of card type is needed; the relevant card ranges must be added to the file.

CardRange.xml can also be used to map cards by range to a suitable description for display on the receipt. CardRange.xml includes comments to explain the layout.

It is anticipated that each POS development team will want to prepare a suitable CardRange.xml for their specific POS requirements, in which case the file can be replaced as required.

Note: For more information, see the *Oracle Retail EFTLink CardRange.xml Guide* available on My Oracle Support (Doc ID 2266221.1) using the following link:

`https://support.oracle.com/rs?type=doc&id=2266221.1`

Logging - EFTLink Framework and Core

EFTLink uses a standard java logging package - `log4j2`. It maintains a daily log file - `eftlink_YYYY-MM-DD.log` - and deletes log files after 30 days. Both the framework and the core log into this file.

Log files are located in the log subdirectory and are created as soon as EFTLink starts. By default, info level logging is enabled. This means that key information is logged but the files are kept as small as possible.

To keep files for longer, or increase the logging level, set `log4j2.xml` appropriately. Edit the `log4j2.xml` configuration file which is located in the main EFTLink directory.

For debug logging change the following entry:

```
<Root level="info">
```

to

```
<Root level="debug">
```

Logging at debug level does not noticeably affect system performance, but does generate larger log files. To retain log files for longer, edit:

```
<Delete basePath="log" maxDepth="1">
    <IfLastModified age="30d" />
</Delete>
```

and alter the age parameter to a number of days to keep files after the current day (default is 30d).

Consider available disk space when choosing a number of days to retain log files.

Multiple log files are configured in the standard `log4j2.xml` configuration file:

- EFTLinkGlobal - contains log information from all sources
- EFTLink - contains log information from the framework

A core may have its own `log4j2.xml` configuration file copied in during install to log to additional files for 3rd party libraries.

After installing EFTLink as a service, then starting the service, the log file will show about 16 lines, with some basic information, and log that it is deferring all initialization until POS type is known. Once a POS starts, you see details of the core started, with the settings used by the core and initialization progress logged, along with subsequent processing data.

In the case of a MultiServerLauncher / PedPooling installation, the standard `log4j2.xml` file requires alteration to include server appenders/loggers. See installation document for further details.

Translation

Translation is the process of interpreting and adapting text from one language into another. Although the code itself is not translated, components of the application that are translated may include the following, among others:

- Graphical user interface (GUI)
- Error messages

The following components are not usually translated:

- Documentation (for example, Online Help, Release Notes, Installation Guide, User Guide, Operations Guide)
- Batch programs and messages
- Log files
- Configuration Tools
- Reports
- Demo data
- Training Materials

Most display messages are generated by the core in use or by the host, in which case they are displayed without change. There are also some display messages generated by EFTLink itself. These are defined in `LangEN.properties`, which is held externally in the root folder of EFTLink - if necessary, the file in the EFTLink root folder can be edited.

The EFTLink framework supports a number of other languages. Setting EFTLink framework to use one of these is in `EftLinkConfig.properties`

```
DisplayLanguage = EN
```

Possible values include:

Table 4–1 Display Language Settings

Language	Setting
Chinese (Simplified)	CN
German	DE
English	EN
Spanish	ES
French	FR
Italian	IT
Japanese	JP
Dutch	NL
Portuguese	PT
Russian	RU
Swedish	SV

Each of these has its own language property file, for example `LangDE.properties`. The file is held in the root EFTLink folder where it can be edited.

> **Note:** The languages that do not use the Latin alphabet have the characters defined in Unicode in their property file. To display messages in Chinese, Japanese or Russian the operating system must support those languages.

Setting the value `DisplayLanguage =`

in `EftlinkConfig.properties` will also control which language a core will use for core specific translations.

Table 4–2 Core Specific Translations

Core	Language Included
Adyen	Chinese (Simplified)
	German
	English
	Spanish
	French
	Italian
	Japanese
	Dutch
	Portuguese
	Russian
	Swedish
AJB FiPay	Chinese (Simplified)
	German
	English
	Spanish
	French
	Italian
	Japanese
	Dutch
	Portuguese
	Russian
	Swedish
Cayan	Chinese (Simplified)
	German
	English
	Spanish
	French
	Italian
	Japanese
	Dutch
	Portuguese
	Russian
	Swedish

Table 4–2 (Cont.) Core Specific Translations

Core	Language Included
Merchant Link	Chinese (Simplified)
	German
	English
	Spanish
	French
	Italian
	Japanese
	Dutch
	Portuguese
	Russian
	Swedish
OPI Retail	Chinese (Simplified)
	German
	English
	Spanish
	French
	Italian
	Japanese
	Dutch
	Portuguese
	Russian
	Swedish
PayPal	Chinese (Simplified)
	German
	English
	Spanish
	French
	Italian
	Japanese
	Dutch
	Portuguese
	Russian
	Swedish

Table 4–2 (Cont.) Core Specific Translations

Core	Language Included
SixPayment Services MPD	Chinese (Simplified)
	German
	English
	Spanish
	French
	Italian
	Japanese
	Dutch
	Portuguese
	Russian
	Swedish
Tender Retail	Chinese (Simplified)
	German
	English
	Spanish
	French
	Italian
	Japanese
	Dutch
	Portuguese
	Russian
	Swedish
The Logic Group SolveConnect	Chinese (Simplified)
	German
	English
	Spanish
	French
	Italian
	Japanese
	Dutch
	Portuguese
	Russian
	Swedish
Verifone Ocius Sentinel	No translation included

Table 4–2 (Cont.) Core Specific Translations

Core	Language Included
Verifone Point US	Chinese (Simplified)
	German
	English
	Spanish
	French
	Italian
	Japanese
	Dutch
	Portuguese
	Russian
	Swedish
World Pay	No translation included

A

Appendix: Installation Order

This section provides a guideline as to the order in which the Oracle Retail applications should be installed. If a retailer has chosen to use some, but not all, of the applications the order is still valid less the applications not being installed.

Note: The installation order is not meant to imply integration between products.

Enterprise Installation Order

1. Oracle Retail Merchandising System (RMS), Oracle Retail Trade Management (RTM)
2. Oracle Retail Sales Audit (ReSA)
3. Oracle Retail Extract, Transform, Load (RETL)
4. Oracle Retail Warehouse Management System (RWMS)
5. Oracle Retail Invoice Matching (ReIM)
6. Oracle Retail Price Management (RPM)
7. Oracle Retail Allocation
8. Oracle Retail Mobile Merchandising (ORMM)
9. Oracle Retail Customer Engagement (ORCE)
10. Oracle Retail Xstore Office
11. Oracle Retail Xstore Point-of-Service, including Xstore Point-of-Service for Grocery, and including Xstore Mobile
12. Oracle Retail Xstore Environment
13. Oracle Retail EFTLink
14. Oracle Retail Store Inventory Management (SIM), including Mobile SIM
15. Oracle Retail Predictive Application Server (RPAS)
16. Oracle Retail Predictive Application Server Batch Script Architecture (RPAS BSA)
17. Oracle Retail Demand Forecasting (RDF)
18. Oracle Retail Category Management Planning and Optimization/Macro Space Optimization (CMPO/MSO)
19. Oracle Retail Replenishment Optimization (RO)

20. Oracle Retail Regular Price Optimization (RPO)
21. Oracle Retail Merchandise Financial Planning (MFP)
22. Oracle Retail Size Profile Optimization (SPO)
23. Oracle Retail Assortment Planning (AP)
24. Oracle Retail Item Planning (IP)
25. Oracle Retail Item Planning Configured for COE (IP COE)
26. Oracle Retail Advanced Inventory Planning (AIP)
27. Oracle Retail Integration Bus (RIB)
28. Oracle Retail Service Backbone (RSB)
29. Oracle Retail Financial Integration (ORFI)
30. Oracle Retail Bulk Data Integration (BDI)
31. Oracle Retail Integration Console (RIC)
32. Oracle Commerce Retail Extension Module (ORXM)
33. Oracle Retail Data Extractor for Merchandising
34. Oracle Retail Clearance Optimization Engine (COE)
35. Oracle Retail Analytic Parameter Calculator for Regular Price Optimization (APC-RPO)
36. Oracle Retail Insights, including Retail Merchandising Insights (previously Retail Merchandising Analytics) and Retail Customer Insights (previously Retail Customer Analytics)
37. Oracle Retail Order Broker

Glossary

Card Circuit

A textual description of the card returned by the payment system, often where the payment system does not return a card IIN

Card IIN

The first few numbers of a card PAN that will identify the card type

IFSF

International Forecourt Standards Forum

DCC

Dynamic Currency Conversion. Converting a sale into the home currency of the card holder by the EFT payment system

JVM

Java Virtual Machine

PED

Pin entry device

PED Pooling

Where the EFTLink Server is used to manage a pool of PEDs to be shared between the POSs and allocated dynamically

Print Pooling

Where the EFTLink Server is used to manage a pool of printers to be shared between the POSs and allocated dynamically

Tender

A description or grouping of a payment type. Sometimes called a MOP (Method of Payment)