Oracle® Retail EFTLink

Framework Installation and Configuration Guide

Release 25.0.0

G34822-02

August 2025

Oracle Retail EFTLink Framework Installation and Configuration Guide, Release 25.0.0

G34822-02

Primary Author: Bernadette Goodman

Contributors: Sean Hamill

Contents

Send Us Your Comments

Oracle Retail EFTLink Framework Installation and 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(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 25.0.0 documentation set:

- *Oracle Retail EFTLink Release Notes*
- *Oracle Retail EFTLink Core Configuration Guide*
- *Oracle Retail EFTLink Framework Advanced Features 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, 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 the downloaded archive file.
2. Either use the binaries or the Installer/Upgrader deliverables to configure your eftlink environment which typically consists of one or more cores to connect to the EFT system or terminal to be used.

The *Oracle Retail EFTLink Framework Installation and Configuration 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 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 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 goods in the retail sector. 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.

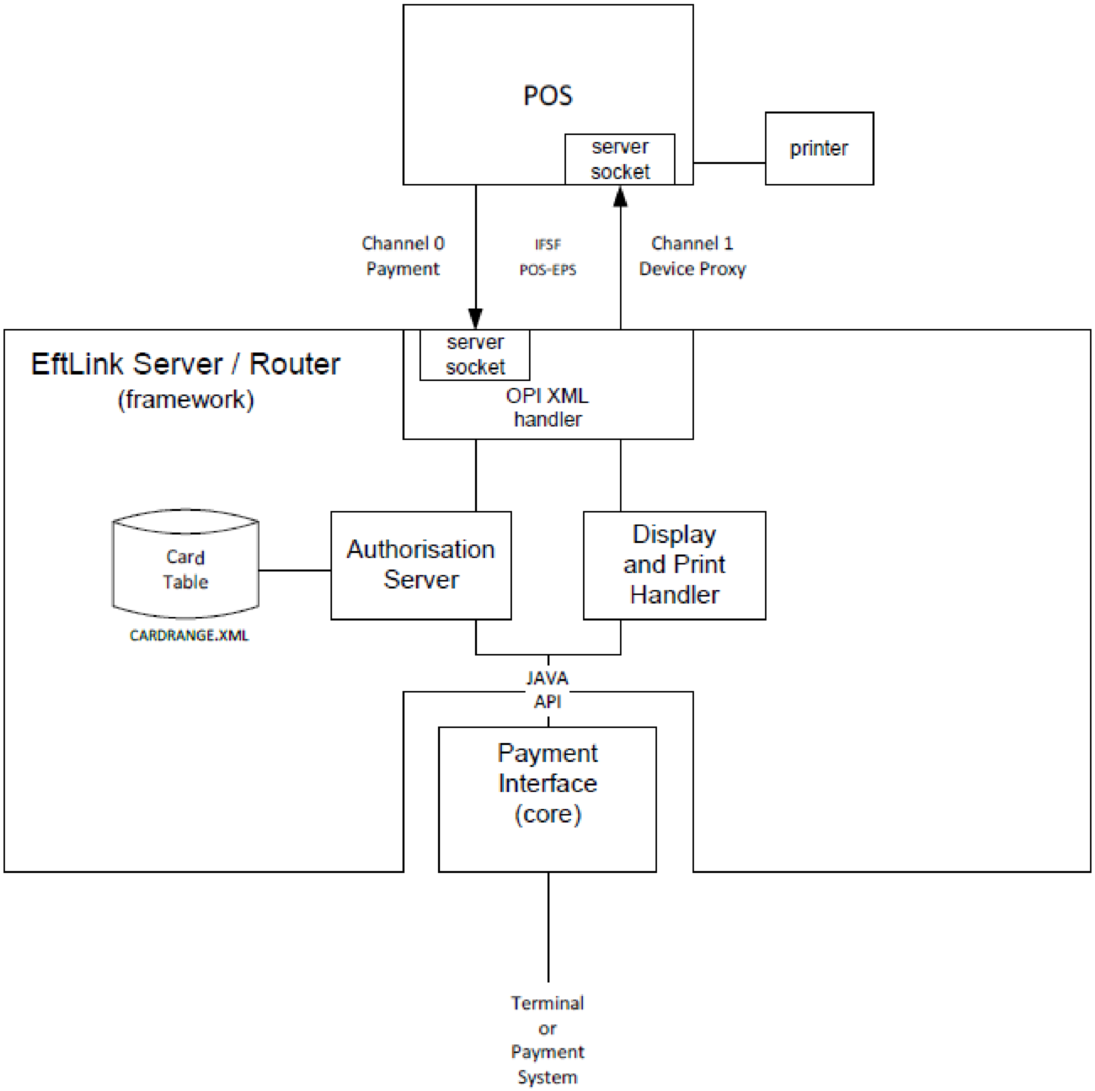
Examples of where EFTLink is used include:

- 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 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 Installation

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

- Skillset Required
- Prerequisites
- Installing EFTLink

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 updates.

The following optional steps are intended for anyone who requires JRE 21 but intend to use an earlier version of Xstore v25.

Using Xstore JRE Packager tool to generate JRE 21 base of JDK 21:

1. From https://edelivery.oracle.com download the Oracle Retail Xstore Point of Service 25.0.0.0.0 (Oracle Retail Xstore Point of Service) relevant to the O/S you wish to run EFTLink with.

2. Extract OracleRetailXstoreCommon_25_0_0_0_XST_0_0_0 and locate the tool jrepackager.
3. From https://www.oracle.com/in/java/technologies/downloads/ download the latest JDK Compressed Archive of JDK 21 and copy the archive into the same folder to where xstore-25.0.0.0.x-0.0.0-XST-jrepackager.jar exists.

   If you are using a Linux WSL environment on a windows machine, you may need to install the following packages to use the jrepackager tool.

   sudo dnf install libXext

   sudo dnf install libXrender

   sudo dnf install libXtst

   sudo dnf install freetype
4. Edit the ant.install.properties file and update the “jdk.package” key with the location of the JDK 21 compress archive file.

   for example: jdk.package=C\:\\DTVINST\jrepackager\\jdk-21.0.6_windows-x64_bin.zip
5. From the terminal run xstore-25.0.0.0.x-0.0.0-XST-jrepackager.jar.

   For example: java -jar C\:\\DTVINST\jrepackager\\xstore-25.0.0.0.x-0.0.0-XST-jrepackager.jar gui

   Note: The java version you're running the jrepackager with must be JDK 21. If you do not provide the “gui” at the end of the above run statement, then the jrepackager will run silently and will look to the ant.install.properties file for its configuration.
6. if successful, the jrepackager will produce a JRE 21 compressed archive. For example, XST-jre-21.0.6-windows_64.zip
7. Extract the compress archive and copy the JRE folder to your preferred location.

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 8
- Oracle Enterprise Linux 9
- Windows POSReady 7
- Windows 10
- Windows 10 IOT Enterprise LTSB 2016 (1607)
- Windows 11

Java

EFTLink framework and all strategic cores will run with any version of Java from 21.

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

Situations may arise where a specific version of the Java JRE is required. For example, running EFTLink may require a different version than the one used by the POS. In such cases, a separate JRE installation might be needed; otherwise, both applications can share the same JRE location. Refer to the *Oracle Retail EFTLink Core Configuration Guide* for any core JRE requirements.

Environment Variable

The EFTLink framework uses an environmental variable called EFTLINK_JAVA_HOME which provides EFTLink Server, EFTLink-Rest-API-Service, and all configuration scripts with the location of the Java JRE. You can either manually add this environment variable, or alternately each script, when executed will call setenv.bat (on Windows) or setenv.sh (on a Linux kernel) automatically setting the environment variable.

The setenv script will check if the Xstore JRE, located at C:\jre (on Windows) or /opt/jre (on a Linux kernel) is compatible, if so, then EFTLINK_JAVA_HOME will be set to the same location as Xstore. If not, it will be set to the alternative location: C:\jre_21 or /opt/jre_21.

Installing EFTLink

- Runnable Installer/Upgrader Jar
- Manual Installation
- Post Installation Steps

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-xx.x-installer.jar` and `eftlink-xx.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.	`C\:\\eftlink` (windows) `/opt/eftlink` (linux)	`C\:\\eftlink`
jreLocation	The location of the JRE to be used.	`C\:\\jre` (windows) `/opt/jre` (linux)	`C\:\\jre_21`
Note: The following settings only apply to the eftlink-25.x-installer.jar and therefore not applicable when running the eftlink-25.x-upgrader.jar.			
ServerChannel0	Configures EFTLink eftlinkConfig.properties ServerChannel0 property setting.	10100	`10100`
ServerChannel1	Configures EFTLink eftlinkConfig.properties ServerChannel1 property setting.	10101	`10101`
installationMode	Installation mode value options are standalone, server, and pedpooling.	standalone	`server`
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	5
coreSetup	coreSetup value options are: singlecore, and multicore.	singlecore	`singlecore`
EPSCore<n>	EPSCore< 0 - 5> 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 = oracle.eftlink.opiretail.OPIRetailCore	`EPSCore0 = oracle.eftlink.opiretail.OPIRetailCore`
CustomFormCore = 0	A core designated to handle custom forms operations. Only used when coreSetup = multicore.	CustomFormCore = 0	
EwalletCore	A core designated to handle EWallet operations. Only used when coreSetup = multicore.	EwalletCore = 0	

Table 2-1 (Cont.) Mandatory Installer Settings

Setting	Description	Default	Example
GiftCardCore	A core designated to handle Gift Card operations. Only used when coreSetup = multicore.	GiftCardCore = 0	
PayByLinkCore	A core designated to handle PayByLink operations. Only used when coreSetup = multicore.	PayByLinkCore = 0	
ReferralCore	A core designated to handle Referrals operations. Only used when coreSetup = multicore.	ReferralCore = 0	

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.
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:\xstoredata\xstore\keys,` or prior to version 22 of Xstore in `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 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.

Manual Installation

This section describes the installation sequence of EFTLink using the binary files.

- Step 1 - Creating the EFTLink Folder
- Step 2 - Install the Files
- Step 3 - Run the Installation Script
- Step 4 - Copy TLS Communication Keys

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, which, 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.
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.
linux	Folder containing files for tanuki wrapper.
linux_64	Folder containing files for tanuki wrapper.
log	Folder containing the log files.
reports	Folder containing the API documentation for the framework.
tmp	Working folder for EFTLink.
windows	

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

Files/Folder	Comment
windows 64	
wrapper	
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	Configuration properties for the eftlink-rest-api service.
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	Jetty configuration script for use in conjunction with the eftlink-rest-api service.
LangCN.properties	Chinese (Simplified) translation file for eftlink framework.
LangDE.properties	German translation file for eftlink framework.
LangEN.properties	English translation file for eftlink framework.
LangES.properties	Spanish translation file for eftlink framework.
LangFR.properties	French translation file for eftlink framework.
LangIT.properties	Italian translation file for eftlink framework.
LangJP.properties	Japanese translation file for eftlink framework.
LangNL.properties	Dutch translation file for eftlink framework.
LangPT.properties	Portuguese translation file for eftlink framework.
LangRU.properties	Russian translation file for eftlink framework.
LangSV.properties	Swedish translation file for eftlink framework.
Log4j2.xml	Log4j2 configuration file.
Setenv.bat	Script file for setting the environment variable EFTLINK_JAVA_HOME.

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

Files/Folder	Comment
Setenv.sh	Script file for setting the environment variable EFTLINK_JAVA_HOME.
truststore.bat	A batch file used to create a truststore to ensure secure communications between EFTLink and EPS terminal.
truststore.sh	A Linux script used to create a truststore to ensure secure communications between EFTLink and EPS terminal.

Step 3 - Run the Installation Script

To set up EFTLink with an active core:

Table 2-3 Core Names

Core Name (Case insensitive)	Description
Cayan	Cayan
FIPay	AJB FIPay
OciusSentinel	Verifone Ocius Sentinel
OPIRetail	OPIRetail
PayByLink	PayByLink (as secondary core only)
PayPal	PayPal (supports Ewallet transactions only)
PointUS	Verifone Point (US)
SixPay	Six Payment Services MPD
SolveConnect	The Logic Group SolveConnect
TenderRetail	TenderRetail
WorldPay	WorldPay

For Windows,

- From the run line type:`<installation directory\installcore.bat` (Advanced setup)
- From a command terminal:`<installation directory\installcore.bat <CoreName>` (Legacy setup)

For Linux,

Open a terminal and change the directory to the EFTLink installation path and type: `installcore.sh <coreName>`

Follow the on-screen instructions. The batch or script file does two things:

- Configures EftlinkConfig.properties with the desired core(s).
- Copies the selected core property file from the specific core folder to the main EFTLink folder, where it will be the active file.
- Installs EFTLink as a Windows Service.
- Creates TLS Communication Keys.

The table below lists the full classpath to the supplied core application.

Table 2-4 Core Classpath

Core	Classpath
AJB FIPay	manito.eft.ajb.FIPayCore
Cayan	manito.eft.cayan.CayanCore
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
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 - Copy TLS Communication Keys

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:\xstoredata\xstore\keys, or prior to version 22 of xstore in C:\xstore\keys).

Post Installation Steps

By default, in Windows, the 'OPI Server' service runs under the Local system account user. To allow the EFTLink service to create dynamic key store files, a user with administrative privileges may be required. To configure this in the services panel, right click on the OPI Server service. Select the **Properties** and then select the **Log on** tab. Select **This account** and input the user's credentials and select **OK**.

- 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.
- Starting from release 24, as mentioned in the *Oracle Retail EFTLink Core Configuration Guide*, the OPIRetail, Cayan, and PayByLink cores enforce the use of a dedicated truststore. Please pay particular attention to the follow section: Securing Communication to EPS Devices by Importing Trusted Certificates.

Altering the Windows Service

By default, EFTLink is install as a window service (OPI Server). Below commands can be run post installation to either alter the services state or remove it altogether.

Windows Configuration

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

- `eftlink.bat console` - run the application with a console.
- `eftlink.bat start` - start eftlink once installed as a Windows service.
- `eftlink.bat restart` - restart eftlink once installed as a Windows service.
- `eftlink.bat stop` - stop eftlink once installed as a Windows service.
- `eftlink.bat install` - install eftlink as a Windows service.
- `eftlink.bat remove` - uninstall eftlink as a Windows service.
- `eftlink.bat help` - show this message.

Linux

`sudo./eftlink.sh stop` - stop eftlink service.

`sudo./eftlink.sh status` - service status.

`sudo./eftlink.sh restart` - restart service.

`sudo./eftlink.sh condrestart` - only starts the daemon if it is currently running.

Altering the Wrapper Configuration

By default, EFTLink allows Tanuki wrapper to manage the deployment.

Please spend time reviewing all configuration files held at "<EFTLink Installation dir>/wrapper/conf/" as under certain configuration you may need to edit these files.

For example:

- By default, EFTLink is configured to use the CardRange.xml to control the card mask applied when sending a PAN. If you are intending to not use the cardrange.xml, then EFTLink will fall back to its default masking rule which can be configured within the eftlink.conf file.

  wrapper.java.additional.<n>= -DEFTLink.UnmaskedPanStartLength=0

  wrapper.java.additional.<n>= -DEFTLink.UnmaskedPanEndLength=4

  The accepted start length range is limited to 0 to 8 whereas the end length range is limited to 0 to 4. The defaults applied are to mask the whole PAN apart from the last four digits.

Securing Communication by Creating TLS Communication Keys

Important: When EFTLink and Xstore are running separate JREs, it is strongly recommended that you update both JREs to their latest versions to avoid issues with TLS Communications.

Although TLS communication Keys are generated by default. You may wish to regenerate your keys. 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> [-dname]

   CreateKeys.sh    -e <algorithm> <bitlength> <signAlgorithm> <daysValidity> [-dname]
   ```

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

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

   Table 2-5   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
	[-dname]	Prompt for POS and Eftlink keystores certificate Distinguished Name information.	

3. Once encryption keys are created, five 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

   `comms.keystore.properties` - required to be held on both POS and EFTLink Server

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:\xstoredata\xstore\keys`, or prior to version 22 of Xstore in `C:\xstore\keys`).

   `pos.private.jks`

   `eftlink.public.jks`

5. The following file should be COPIED 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:\xstoredata\xstore\keys`, or prior to version 22 of Xstore in `C:\xstore\keys`):

```
comms.keystore.properties
```

6. This will leave the following three files on the EFTLink server in the folder [eftlink root] \keys:

   ```
   eftlink.private.jks
   ```

   ```
   pos.public.jks
   ```

   ```
   comms.keystore.properties
   ```

7. 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.

8. 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 like those when used to generate self-signed certificates but specify the first parameter as -s.

   ```
   CreateKeys.bat    -s <algorithm> <bitlength> <signAlgorithm> <daysValidity> [-dname]
   ```

   ```
   CreateKeys.sh    -s <algorithm> <bitlength> <signAlgorithm> <daysValidity> [-dname]
   ```

   For example,

   ```
   CreateKeys.bat-s RSA 4096 SHA256withRSA 750
   ```

   ```
   CreateKeys.bat-s RSA 4096 SHA256withRSA 750 -dname
   ```

Table 2-6   CA Certificate Parameters

Switch	Parameter	Description	Supported Value
-s	<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)

Table 2-6 (Cont.) CA Certificate Parameters

Switch	Parameter	Description	Supported Value
	<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
	[-dname]	Prompt for POS and Eftlink keystores certificate Distinguished Name information.	

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

   `comms.keystore.properties` - keystore encryption data 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 2-7 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:\xstoredata\xstore\keys`, or prior to version 22 of Xstore in `C:\xstore\keys`):

   `pos.private.jks`

   `eftlink.public.jks`

8. The following file should be COPIED 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:\xstoredata\xstore\keys`, or prior to version 22 of Xstore in `C:\xstore\keys`):

   `comms.keystore.properties`

9. This will leave the following three files on the EFTLink server in the folder [eftlink root]\keys:

   `eftlink.private.jks`

   `pos.public.jks`

   `comms.keystore.properties`

10. 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 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 version 20 onwards, expiry of TLS certificates is enforced by default. Self-signed certificates will be valid for a maximum of 750 days.

11. 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.

Securing Communication to EPS Devices by Importing Trusted Certificates

Tip: Please note if you alter the Java JRE used to run EFTLink away from its defaults, please be aware that you may also have to edit the "truststore" script to point to the new location of the JRE.

Only for cores that support trusted connection to EPS devices, currently just OPIRetail core v24, a truststore with the trusted certificates must be created. A batch file, truststore.bat, and a Linux script, truststore.sh, are included in the EFTLink project to facilitate the creation of the truststore.

1. Locate the `truststore.bat / truststore.sh` file in the EFTLink folder.
2. From a terminal, run the truststore script file with an appropriate set of parameters to import the trusted EPS device certificate chain.

   ```
   truststore.bat -i <alias> <certificate file>
   ```

   ```
   truststore.sh -i <alias> <certificate file>
   ```

   For example: `truststore.bat -i providername provider_trusted_public_cert.crt`
3. Repeat step 2 for every public trusted certificate or part of the chain that need to be imported.

Table 2-8 Import of Trusted Certificate Parameters

Switch	Parameter	Description	Supported Value
-i	<alias>	Alias for the certificate.	Alphanumeric, must be unique for every imported certificate.
	<certificate file>	Public trusted certificate file.	EPS device security certificate chain in PEM format. **Consult the payment provider's instructions to obtain the correct certificate used by the trusted EPS device for production or test environment**.

Once the first trusted certificate will be imported, the following files will be created on the system in the keys subfolder of EFTLink:

`eftlink.truststore.jks` required to be held on the EFTLink Server

`eftlink.truststore.properties` required to be held on EFTLink Server

A list of all certificates imported on the truststore can be obtained with:

```
truststore.bat -l
```

Eventually a certificate can be removed from the truststore with:

```
truststore.bat -d alias
```

Prints a certificate file content:

```
truststore.bat -p certificate_file.crt
```

Deploying EFTLink within a Docker Container

Build Your Docker Image

1. Create a working directory.

   For example:

   ```
   mkdir $HOME/docker-build-files
   ```

2. Extract the contents from container.zip to your working directory.
3. Copy the EFTLink binaries zip file (v<xx>.0.0.<nnn>.binaries.zip) to your working directory.
4. Copy the XST-jre-21.0.<XX>-linux_64.zip (from jrepackager) to your working directory.
5. Run the following command from the command shell within your working directory:

```
docker build \

--build-arg eftlink_zip=v<xx>.0.0.<nnn>.zip \

--build-arg jre_zip=XST-jre-21.0.<nnn>-linux_64.zip \

--build-arg eftlink_user_uid=<user uid> \

--build-arg eftlink_user_gid=<user gid> \

-f Dockerfile \

-t eftlink:latest .
```

For example:

```
docker build \

--build-arg eftlink_zip=v25.0.0.298.binaries.zip \

--build-arg jre_zip=XST-jre-21.0.12-linux_64.zip \

--build-arg eftlink_user_uid=9090 \

--build-arg eftlink_user_gid=9090 \

-f Dockerfile \

-t eftlink:latest .
```

This will build your Docker image. Once the process has finished, you can see the completed image by running:

```
docker image ls eftlink
```

Output:

```
REPOSITORY TAG IMAGE ID CREATED SIZE

eftlink latest 3a0e731911b4 6 hours ago 596MB
```

Running in Docker (Standalone)

You will need to decide which configuration file need to be made persistent (live outside of the docker container).

To do this, create a working directory. For example:

```
mkdir $HOME/eftlink-config
```

Copy the desire files into this directory and configure them for your intended environment.

The recommended files and folders to persist are as follows:

- tmp
- keys
- log
- EftlinkConfig.properties
- eftlink-rest-api.properties
- jetty.xml
- <core>.properties
- Lang<XX>_<core>.properties

Command example which will run a container with simple bind mounts:

```
docker run --name eftlink --user eftlink --publish 10100:10100 --publish 8443:8443 /

--volume <path>/tmp:/opt/eftlink/tmp /

--volume <path>/keys:/opt/eftlink/keys

--volume <path>/log:/opt/eftlink/log /

--volume <path>/EftlinkConfig.properties:/opt/eftlink/EftlinkConfig.properties /

--volume <path>/eftlink-rest-api.properties:/opt/eftlink/eftlink-rest-api.properties /

--volume <path>/jetty.xml:/opt/eftlink/jetty.xml /

--volume <path>/<core>.properties:/opt/eftlink/<core>.properties /

--volume <path>/Lang<XX>_<core>.properties:/opt/eftlink/Lang<XX>_<core>.properties /

--tty --rm --interactive --workdir /opt/eftlink eftlink
```

For example:

```
docker run --name eftlink --user eftlink --publish 10100:10100 --publish 8443:8443 /

--volume <path>/tmp:/opt/eftlink/tmp /

--volume <path>/keys:/opt/eftlink/keys

--volume <path>/log:/opt/eftlink/log /

--volume <path>/EftlinkConfig.properties:/opt/eftlink/EftlinkConfig.properties /

--volume <path>/eftlink-rest-api.properties:/opt/eftlink/eftlink-rest-api.properties /

--volume <path>/jetty.xml:/opt/eftlink/jetty.xml /
```

--volume <path>/opiretail.properties:/opt/eftlink/opiretail.properties /

--volume <path>/LangEN_OPIRetail.properties:/opt/eftlink/LangEN_OPIRetail.properties /

--tty --rm --interactive --workdir /opt/eftlink eftlink

To confirm that the container has started, you check the logs file in your working directory or enter:

docker ps | grep <container_name_or_id>

For example:

docker ps | grep eftlink

64c8f733f45e eftlink "/opt/eftlink/startu…" 7 minutes ago Up 7 minutes (healthy) 0.0.0.0:8443->8443/tcp, :::8443->8443/tcp, 0.0.0.0:10100->10100/tcp, :::10100->10100/tcp, 10101-10200/tcp eftlink

Running EFTLink in Docker Swarm

If you want to run EFTLink within a Docker Swarm, follow these steps:

1. Create a Docker swarm:

   docker swarm init --advertise-addr <IP ADDRESS>

2. Confirm the machine is now part of the swarm by typing:

   docker node ls

3. Create a virtual network within the swarm:

   docker network create --scope swarm --driver overlay xst

4. Create a data volume:

   For example:

   docker volume create eftlink-reg1

5. Create a config.

   Docker Configs are where we store config files and other static data.

   The recommended files are:

   - comms.keystore.properties
   - eftlink.private.jks
   - eftlink.public.jks
   - pos.private.jks
   - pos.public.jks
   - eftlink.truststore.jks
   - eftlink.truststore.properties
   - eftlink-rest-api.keystore
   - eftlink-rest-api.properties
   - jetty.xml
   - eftlink-rest-api-log4j2.xml
   - log4j2.xml

- EftlinkConfig.properties
- <core>.properties
- Lang<XX>_<core>.properties

6. Add these to the config by running through the files you wish to add:

   ```
   docker config create <descriptive_name> <file_location>
   ```

   For example:

   ```
   docker config create eftlink-reg1-comms.keystore.properties ./keys/comms.keystore.properties
   docker config create eftlink-reg1-eftlink.private.jks ./keys/eftlink.private.jks
   docker config create eftlink-reg1-eftlink.public.jks ./keys/eftlink.public.jks
   docker config create eftlink-reg1-pos.private.jks ./keys/pos.private.jks
   docker config create eftlink-reg1-pos.public.jks ./keys/pos.public.jks
   docker config create eftlink-reg1-eftlink.truststore.jks ./keys/eftlink.truststore.jks
   docker config create eftlink-reg1-eftlink.truststore.properties ./keys/eftlink.truststore.properties
   docker config create eftlink-reg1-eftlink-rest-api.keystore ./keys/eftlink-rest-api.keystore
   docker config create eftlink-reg1-eftlink-rest-api.properties eftlink-rest-api.properties
   docker config create eftlink-reg1-jetty.xml jetty.xml
   docker config create eftlink-reg1-eftlink-rest-api-log4j2.xml eftlink-rest-api-log4j2.xml
   docker config create eftlink-reg1-log4j2.xml log4j2.xml
   docker config create eftlink-reg1-EftlinkConfig.properties EftlinkConfig.properties
   docker config create eftlink-reg1-opiretail.properties opiretail.properties
   docker config create eftlink-reg1-LangEN_OPIRetail.properties LangEN_OPIRetail.properties
   ```

7. Run to "docker config ls" to display the config, then docker config inspect <Name> to view each file data (Base 64 encoded).

8. Run the stack.

   Below is an example content for a docker-compose-stack.yml. Create the file and copy in everything between the ----

   Remember, YAML files are indentation specific. Once you have created the file run:

   ```
   docker stack deploy -c docker-compose-stack.yml Eftlink-Example
   ```

9. To list the service in the stack, run:

   ```
   docker stack services Eftlink-Example
   ```

   output:

   ```
   ID NAME MODE REPLICAS IMAGE PORTS
   i43oqdgpwmch Eftlink-Example_eftlink-reg1 replicated 1/1 eftlink:latest
   ```

   ```
   ----
   version: "3.9"
   services:
   ```

```
eftlink-reg1:
hostname: reg1-eftlink
image: eftlink:latest
ports:
- target: 10100
published: 10100
protocol: tcp
mode: host
- target: 10101
published: 10101
protocol: tcp
mode: host
- target: 8443
published: 8443
protocol: tcp
mode: host
- target: 8444
published: 8444
protocol: tcp
mode: host
networks:
- xst
deploy:
replicas: 1
volumes:
- eftlink-log:/opt/eftlink/log
configs:
- source: eftlink-reg1-comms.keystore.properties
target: /opt/eftlink/keys/comms.keystore.properties
uid: '9090'
gid: '9090'
mode: 0440
- source: eftlink-reg1-eftlink.private.jks
target: /opt/eftlink/keys/eftlink.private.jks
uid: '9090'
gid: '9090'
mode: 0440
- source: eftlink-reg1-eftlink.public.jks
```

```
target: /opt/eftlink/keys/eftlink.public.jks
uid: '9090'
gid: '9090'
mode: 0440
- source: eftlink-reg1-pos.private.jks
target: /opt/eftlink/keys/pos.private.jks
uid: '9090'
gid: '9090'
mode: 0440
- source: eftlink-reg1-pos.public.jks
target: /opt/eftlink/keys/pos.public.jks
uid: '9090'
gid: '9090'
mode: 0440
- source: eftlink-reg1-eftlink.truststore.jks
target: /opt/eftlink/keys/eftlink.truststore.jks
uid: '9090'
gid: '9090'
mode: 0440
- source: eftlink-reg1-eftlink.truststore.properties
target: /opt/eftlink/keys/eftlink.truststore.properties
uid: '9090'
gid: '9090'
mode: 0440
- source: eftlink-reg1-eftlink-rest-api.keystore
target: /opt/eftlink/keys/eftlink-rest-api.keystore
uid: '9090'
gid: '9090'
mode: 0440
- source: eftlink-reg1-eftlink-rest-api.properties
target: /opt/eftlink/eftlink-rest-api.properties
uid: '9090'
gid: '9090'
mode: 0440
- source: eftlink-reg1-jetty.xml
target: /opt/eftlink/jetty.xml
uid: '9090'
gid: '9090'
```

```
mode: 0440
- source: eftlink-reg1-eftlink-rest-api-log4j2.xml
target: /opt/eftlink/eftlink-rest-api-log4j2.xml
uid: '9090'
gid: '9090'
mode: 0440
- source: eftlink-reg1-log4j2.xml
target: /opt/eftlink/log4j2.xml
uid: '9090'
gid: '9090'
mode: 0440
- source: eftlink-reg1-EftlinkConfig.properties
target: /opt/eftlink/EftlinkConfig.properties
uid: '9090'
gid: '9090'
mode: 0440
- source: eftlink-reg1-opiretail.properties
target: /opt/eftlink/opiretail.properties
uid: '9090'
gid: '9090'
mode: 0440
- source: eftlink-reg1-LangEN_OPIRetail.properties
target: /opt/eftlink/LangEN_OPIRetail.properties
uid: '9090'
gid: '9090'
mode: 0440

volumes:
eftlink-tmp:
eftlink-keys:
eftlink-log:
external: true

configs:
eftlink-reg1-comms.keystore.properties:
external: true
eftlink-reg1-eftlink.private.jks:
external: true
eftlink-reg1-eftlink.public.jks:
```

```
external: true
eftlink-reg1-pos.private.jks:
external: true
eftlink-reg1-pos.public.jks:
external: true
eftlink-reg1-eftlink.truststore.jks:
external: true
eftlink-reg1-eftlink.truststore.properties:
external: true
eftlink-reg1-eftlink-rest-api.keystore:
external: true
eftlink-reg1-eftlink-rest-api.properties:
external: true
eftlink-reg1-jetty.xml:
external: true
eftlink-reg1-eftlink-rest-api-log4j2.xml:
external: true
eftlink-reg1-log4j2.xml:
external: true
eftlink-reg1-EftlinkConfig.properties:
external: true
eftlink-reg1-opiretail.properties:
external: true
eftlink-reg1-LangEN_OPIRetail.properties:
external: true

networks:
xst:
external: true
----
```

10. Clean up commands:

- Stop the stack, run:

  ```
  docker stack rm Eftlink-Example
  ```

- Prune all stopped containers, networks, dangling images and build cache, run:

  ```
  docker system prune -f
  ```

- Prune all volumes

  ```
  docker volume prune -f
  ```

- Prune all images (forcing any intermediate images to need to be re-downloaded) *docker image prune -a*

EFTLink Advanced Configuration Features

See the *Oracle Retail EFTLink Framework Advanced Features Guide* located on OHC and refer to the chapter for the specific feature enrichment.

3

EFTLink Configurable Properties

This chapter describes the EFTLink properties:

- Configuration Settings
- Key Settings
- Secondary Settings

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 = oracle.eftlink.opiretail.OPIRetailCore`
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`
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`

Table 3-2 (Cont.) Secondary Settings

Setting	Description	Default	Example
truststore.ExpiryWarningLogDays	Number of Days prior to trusted certificate expiry that error warnings will be placed in EFTLink log files.	90	`truststoreExpiryWarningLogDays = 90`
OPIServerDelegate	Allows the OPIServer operation to be delegated to an alternate class		`OPIServerDelegate = manito.eft.tlog.TLogOPIServer`
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`

Table 3-2 (Cont.) Secondary Settings

Setting	Description	Default	Example
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`
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`

Table 3-2 (Cont.) Secondary Settings

Setting	Description	Default	Example
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).
ProtocolsWhiteList	Restricts the protocols which are permissible in the connection between POS and EFTLink Server. Default only allows for TLS 1.3 security.	TLSv1.3,TLSv1.2	`ProtocolsWhiteList=TLSv1.3,TLSv1.2`

Table 3-2 (Cont.) Secondary Settings

Setting	Description	Default	Example

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_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	`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`

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

PosType	POS type that EftLink is connected to. This can be set explicitly (for example, 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`

Table 3-2 (Cont.) Secondary Settings

Setting	Description	Default	Example
PosIfsfCompliance	The level of IFSF compliance for the POS interface - IFSF or LUCAS.	IFSF	`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`
DeviceEvents	Whether device events such as CardInserted are supported by the POS. Default false.	false	`DeviceEvents = false`

Table 3-2 (Cont.) Secondary Settings

Setting	Description	Default	Example
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`

Table 3-2 (Cont.) Secondary Settings

Setting	Description	Default	Example

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`
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`

Table 3-2 (Cont.) Secondary Settings

Setting	Description	Default	Example

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`
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`

Table 3-2 (Cont.) Secondary Settings

Setting	Description	Default	Example
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`
DeviceRequestInPedPoolModeEnabled	If true, then EFTLink will accept and route custom form / deviceRequest messages coming from a register to the desired server instance specified within the pos.subpool list.	false	`DeviceRequestInPedPoolModeEnabled = true`
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`
SystemInformationLoggingEnabled	Enable logging of system information at startup.	True	`SystemInformationLoggingEnabled = true`
EnableEventLogDeviceRequests	Enable the sending of critical events to the POS.	True	`EnableEventLogDeviceRequests = true`

Table 3-2 (Cont.) Secondary Settings

Setting	Description	Default	Example
communications.keystore.iterations	Specify number of iterations. Valid range 10000 to 100000	10000	`communications.keystore.iterations = 10000`
communications.keystore.hashbytesize	Specify hash byte size. Valid values 256, 384, 512	384	`communications.keystore.hashbytesize = 384`
communications.keystore.digest	Specify digest Valid values SHA-256, SHA-384, SHA-512	SHA-512	`communications.keystore.digest = SHA-512`
communications.keystore.secretkeyspec	Specify keyspec, currently only AES supported.	AES	`communications.keystore.secretkeyspec = AES`
communications.keystore.factoryinstance	Specify factory, currently only PBKDF2WithHmacSHA512 supported.	PBKDF2WithHmacSHA512	`communications.keystore.factoryinstance = PBKDF2WithHmacSHA512`
communications.keystore.keystoretype	Specify Keystore type. Currently only JKS is supported.	JKS	`communications.keystore.keystoretype = JKS`
truststore.keystore.iterations	Specify Truststore number of iterations. Valid range 10000 to 100000	10000	`truststore.keystore.iterations = 10000`
truststore.keystore.secretkeyspec	Specify Truststore keyspec, currently only AES supported.	AES	`truststore.keystore.secretkeyspec = AES`
truststore.keystore.factoryinstance	Specify Truststore factory, currently only PBKDF2WithHmacSHA512 supported.	PBKDF2WithHmacSHA512	`truststore.keystore.factoryinstance = PBKDF2WithHmacSHA512`
truststore.keystore.hashbytesize	Specify Truststore hash byte size. Valid values 256, 384, 512	384	`truststore.keystore.hashbytesize = 384`
ContentMaskList	Comma delimited list of XML fields to be masked on log.		`ContentMaskList= StoreName, StoreAddress1`

Table 3-2 (Cont.) Secondary Settings

Setting	Description	Default	Example
MaskTextLinesOnDeviceRequest	Determines whether the contents of textline elements within a raised device request are masked within the log file.	true	`MaskTextLinesOnDeviceRequest = false`
OriginalPSPNamesToBypassPSPChecking	Comma delimited list of EPSCore names to skip for checking that takes place on follow on transaction (Voids and Referenced Refunds).		`OriginalPSPNamesToBypassPSPChecking= Simulated, oracle.eftlink.opiretail.OPIRetailCore`

> **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:

- Pan Masking Rules
- Tender Mapping
- Logging - EFTLink Framework and Core
- Translation

PAN Masking Rules

By default, EFTLink is configured to use the CardRange.xml to control the card mask to be applied when sending a PAN. If you are intending not use the cardrange.xml, then EFTLink will fall back to its default masking rule which is to show only the last four digits of the PAN.

This can be altered either by using a cardrange.xml file or by JVM system properties `EFTLink.UnmaskedPanStartLength` and `EFTLink.UnmaskedPanEndLength`.

The accepted start length range is limited to 0 to 8 whereas end length range is limited to 0 to 4. The defaults applied are to mask the whole PAN apart from the last four digits.

The supplied CardRange.xml has been altered to only send the last four digits of PAN by default. Please refer to Partner Enablement *Oracle Retail EFTLink CardRange.xml Guide* on how to make changes.

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`

For eWallet tenders, EFTLink supports the following codes on CSRequest.WalletData.WalletType field:

Table 4-1 Supported eWallet Tenders

Code	Description
ACIPAYAFTER	ACIPayAfter
AFFIRM	Affirm
AFRICAN_EWALLETS	African eWallets
AFTERPAY	AfterPay
AIRTEL	Airtel
ALIPAY	AliPay
ALLIED_WALLET	Allied Wallet
AMAZON_PAY	Amazon Pay
AME	AME
ANDROID_PAY	Android Pay
APPLE_PAY	Apple Pay
ATOME	Atome
CHASE_PAY	Chase Pay
DANA	DANA
DWOLLA	Dwolla
FOURALL	4All
GCASH	GCash
GENERIC_EWALLET	Generic eWallet
GOOGLE_PAY	Google Pay
GRAB_PAY	GrabPay
IZPAY	Izpay
KAKAO_PAY	KakaoPay
KLARNA	Klarna
LAYBUY	LayBuy
LINE	Line
LYF_PAY	Lyf Pay
MASTERPASS	MasterPass
MERCADO_PAGO	Mercado Pago
MOBIKWIK	MobiKwik
MOBILEPAY	MobilePay
MOMO_WALLET	MoMo Wallet
OPENPAY	OpenPay
PAGSEGURO	PagSeguro
PAY_BY_BANK_APP	Pay By Bank app
PAYBACK	PayBack
PAYLIB	Paylib
PAYMAYA	PayMaya

Table 4-1 (Cont.) Supported eWallet Tenders

Code	Description
PAYPAL	PayPal
PAYTM	Paytm
PAYUNIQUE	PayUnique
PICPAY	PicPay
PIX	PIX
QUADPAY	QuadPay
RESERVE_1	Reserve 1
RESERVE_2	Reserve 2
RESERVE_3	Reserve 3
RESERVE_4	Reserve 4
RESERVE_5	Reserve 5
RESERVE_6	Reserve 6
RESERVE_7	Reserve 7
RESERVE_8	Reserve 8
RESERVE_9	Reserve 9
RIACHUELO	Riachuelo
SAMSUNG_PAY	Samsung Pay
SEMPARAR	SemParar
SEQR	SEQR
SEZZLE	Sezzle
SPLITIT	SplitIt
SWISH	Swish
TAPAGO	TáPago
TICKETLOG	TicketLog
TROCO_SIMPLES	Troco Simples
TWINT	Twint
UNKNOWN	Unknown
VEEDIGITAL	VeeDigital
VENMO	Venmo
VIPPS	Vipps
WECHAT_PAY	WeChat Pay
YOYO_WALLET	Yoyo Wallet
ZIP	Zip

Logging - EFTLink Framework and Core

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

Log files are 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 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 several days to keep files after the current day (default is 30d).

Consider available disk space when choosing several days to retain log files.

Log file is configured in the standard `log4j2.xml` configuration file:

- EFTLinkGlobal - contains log information from all sources

A core may have its own `log4j2.xml` configuration file copied in during install to log to additional files for third-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 the 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 several other languages. Setting EFTLink framework to use one of these is in `EftLinkConfig.properties`

```
DisplayLanguage = EN
```

Possible values include:

Table 4-2 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 4-3   Core Specific Translations

Core	Language Included
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
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 4-3 (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 4-3 (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)