Oracle® Retail EFTLink

Framework Advanced Features Guide

Release 21.0
F61680-02
August 2022

Oracle Retail EFTLink Framework Advanced Features Guide, Release 21.0

F61680-02

Primary Author: Tracy Gunston

Contributors: Matthew Preston, Ian Williams

Contents

3 Dynamic Configuration

4 Multiple Cores

Send Us Your Comments

Oracle Retail EFTLink Framework Advanced Features Guide, Release 21.0

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 Advanced Features Guide* describes the advanced features of Oracle Retail EFTLink framework.

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 21.0 documentation set:

- *Oracle Retail EFTLink Release Notes*
- *Oracle Retail EFTLink Core Configuration Guide*
- *Oracle Retail EFTLink Framework Installation and Configuration Guide*
- *Oracle Retail EFTLink Security Guide*
- *Oracle Retail EFTLink Rest API Guide*
- *Oracle Retail EFTLink Xstore Compatibility Guide*
- *Oracle Retail EFTLink Validated Partners Guide*
- *Oracle Retail EFTLink Validated OPI Partners Guide*

Customer Support

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

https://support.oracle.com

When contacting Customer Support, please provide the following:

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

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 21.0) or a later patch release (for example, 21.0.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 Introduction

After installing EFTLink (instructions can be found in the *Oracle Retail EFTLink Framework Installation and Configuration Guide*), you may want to use some of the advanced features of the Framework.

This guide consists of separate chapters for each feature of the Framework; go to the pertinent section for more information. The following cores are supported:

- Dynamic Configuration
- Multiple Cores

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
Server Mode

This chapter describes the configuration of running EFTLink within Server mode.

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 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.

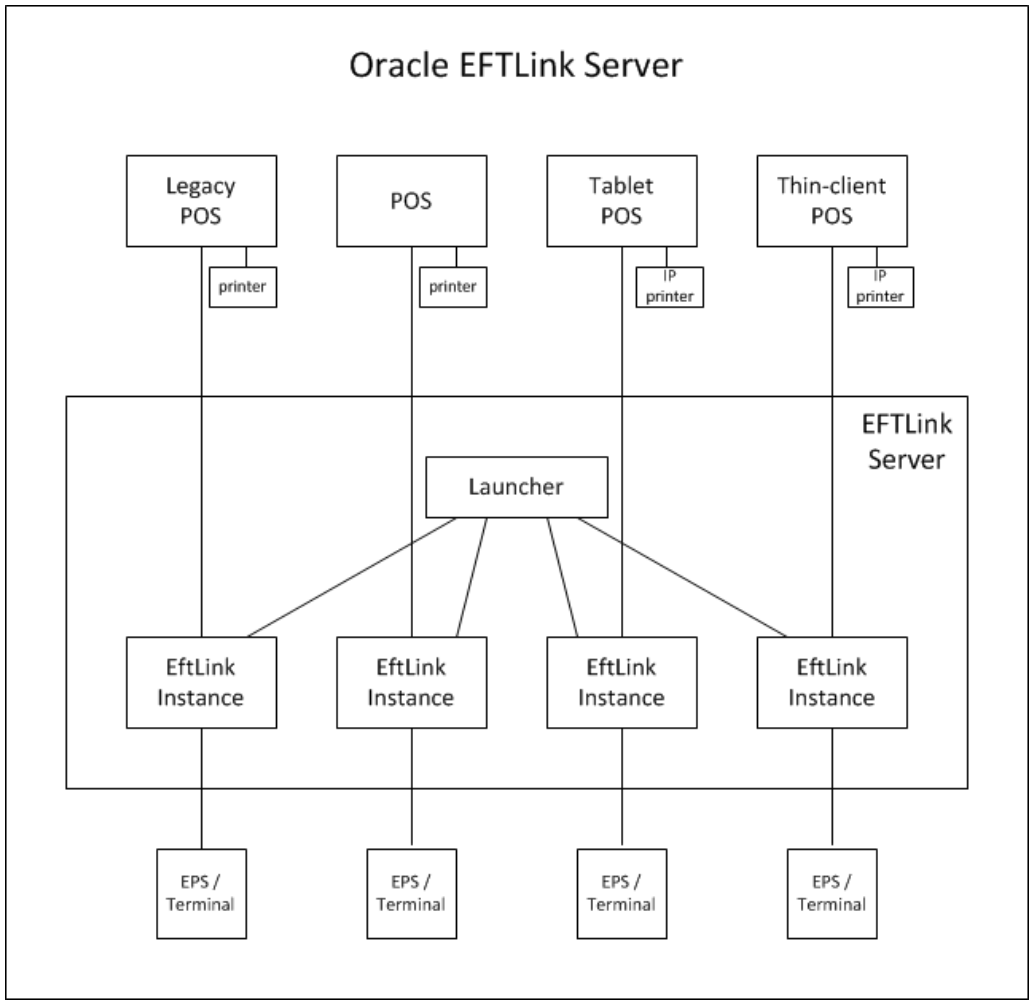
The standard `EFTLinkConfig.properties` will ensure EFTLink is configured for use as an EFTLink Server.

- EFTLink Server - Remote Mode
- MultiJVM
- EFTLink Server - PEDPool Remote Mode
- Configuring 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 2-1 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.

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.

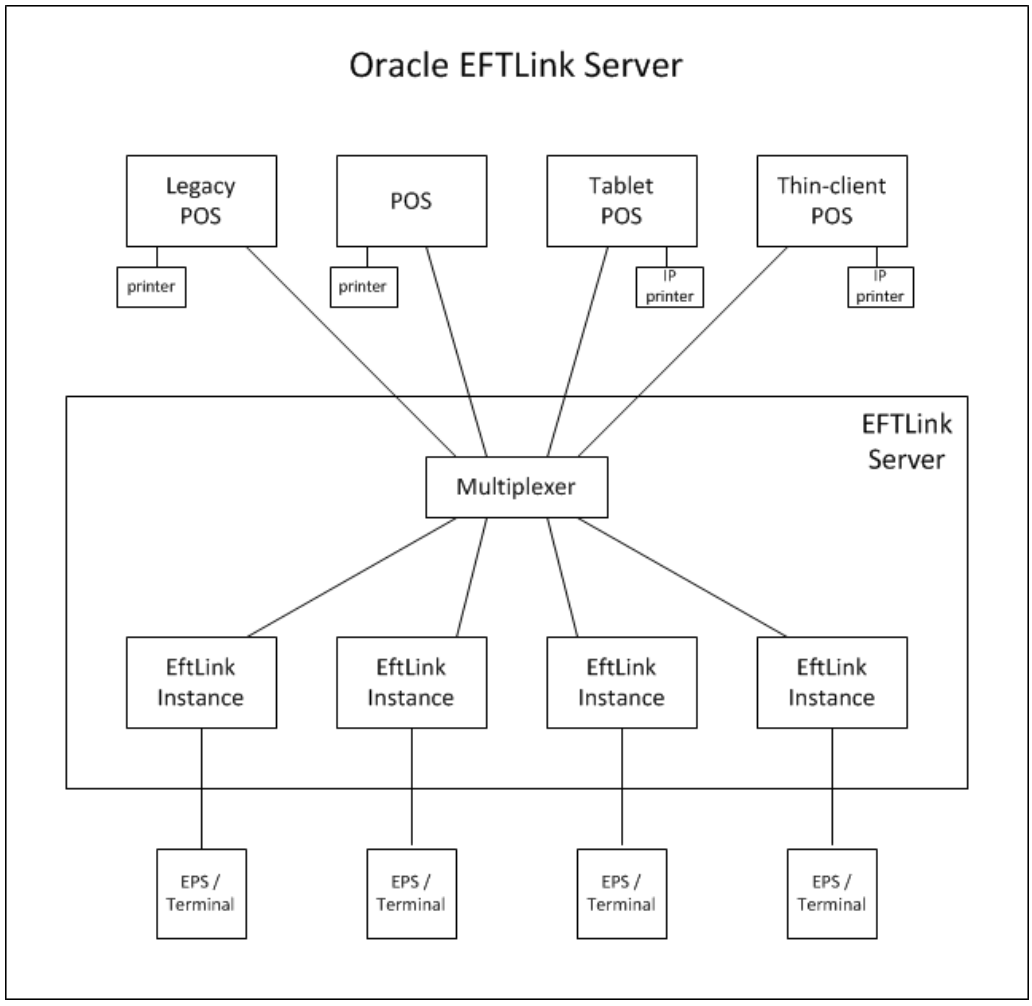
For each server defined under NumServers; EFTLink looks for a corresponding server folder. For example, if NumServers is set to 3, EFTLink looks for server folders named server1, server2 and server3 under the EFTLink directory. These folders must contain their own configuration files, that is; EftLinkConfig.properties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

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

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 2-2 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 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. Each POS must 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 must 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
Dynamic Configuration

This chapter describes the concept and the configuration of running EFTLink Dynamic Cores.

Dynamic configuration is controlled via the EFTLinkConfig.properties setting **DynamicConfiguration** (true by default) and can either be set to true or false.

This allows a POS to override the configuration of EFTLink and any active core that would normally receive their instruction from their respective properties files. This is achieved by the POS sending through an Administrative Service request. There are five sub types of Administrative Service request, these are detailed in the table below:

Table 3-1 Administration Request Sub Types

Setting	Description	Notes
Capability	For any initialized core, a capability request can be sent to check whether that core supports the transaction types.	Supported Transaction type that can be checked are Manual PAN, Loyalty, Reversal, and Cancellation requests.
GetConfig	For any initialized core, this request can be sent to retrieve the current configuration settings for an active core.	Can be useful in conjunction with **Uninitialise** request. For example, **Initialise** are core, gather configuration, **Uninitialise** and then **Initialise** core again with new setting sent within the **Initialise** request.
Initialise	This request must declare the EPSCore(s) that are to be initialized. If no EPSCore is declared, then EFTLink will revert to the `EftlinkConfig.properties` file held in the installation root for its configuration. If no properties are declared within the service request that declares which EPSCores(s) are initialized then EFTLink will revert to defaults.	An **Initialise** request must be sent before a logon request. If an **Initialise** request is sent specifying an EPSCore, then EftlinkConfig.properties is not used and all properties that deviate from defaults setting must come from within the request. Any properties specified inside an EPSCore element is targeted to that core only whereas any properties specified outside are used for all specified cores declared in the service request. Core properties are read in from their respective property files. However, any properties specified within this request will take precedence.
Shutdown	Instructs EFTLink to shut down.	Please note that this terminates EFTLink and therefore EFTLink will not be able to acknowledge nor respond back to the request.

Table 3-1 (Cont.) Administration Request Sub Types

Setting	Description	Notes
Uninitialise	Closes all Initialise cores.	If a core is initialized, then the core configuration is static. If core needs reconfiguring, then this is possible by sending an **Uninitialise** request followed by an **Initialise** request which declares the new configuration changes.

Figure 3-1 EFTLink and Core Initialized through the reading in of Property Files followed by Capability Request

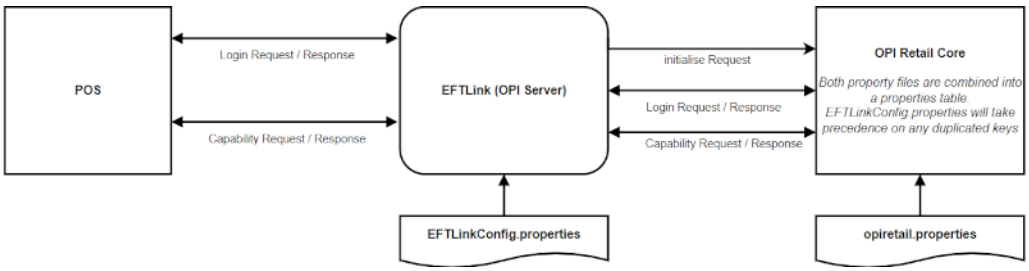
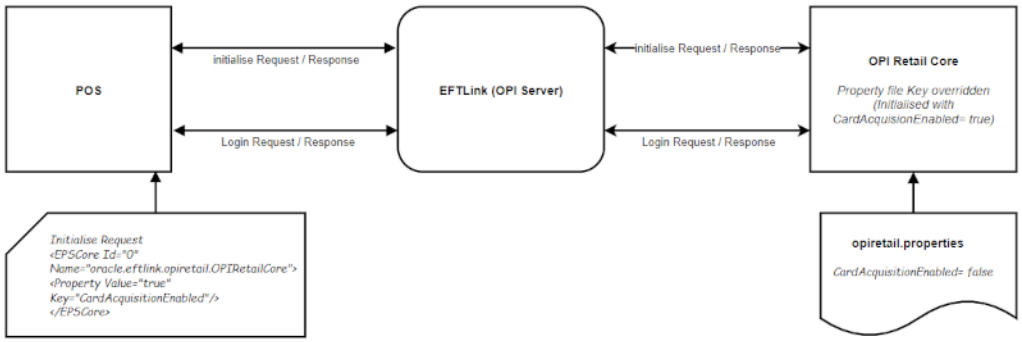


Figure 3-2 EFTLink and Core initialized via POS initialise Server Request



Example Administrative Service Requests and Responses

Capability

Capability Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceRequest RequestID="2" WorkstationID="Workstation: 10100"
ApplicationSender="EFTLink Load Tester" RequestType="Administration"
RequestSubType="Capability">
  <PrivateData>
    <Property Key="Cancellation"/>
    <Property Key="Reversal"/>
    <Property Key="ManualPAN"/>
    <Property Key="Loyalty"/>
  </PrivateData>
  <POSdata LanguageCode="en">
    <POSTimeStamp>2021-08-19T14:03:52</POSTimeStamp>
  </POSdata>
</ServiceRequest>
```

Capability Response (single active core)

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceResponse RequestID="2" WorkstationID="Workstation: 10100"
OverallResult="Success" ApplicationSender="EFTLink Load Tester"
RequestType="Administration" RequestSubType="Capability">
  <PrivateData>
      <Property Value="false" Key="Loyalty"/>
      <Property Value="true" Key="Reversal"/>
      <Property Value="false" Key="ManualPAN"/>
      <Property Value="true" Key="Cancellation"/>
  </PrivateData>
</ServiceResponse>
```

Capability Response (multiple active core)

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceResponse RequestID="2" WorkstationID="Workstation: 10100"
OverallResult="Success" ApplicationSender="EFTLink Load Tester"
RequestType="Administration" RequestSubType="Capability">
  <PrivateData>
    <Core0 CoreName="oracle.eftlink.opiretail.OPIRetailCore">
      <Property Value="true" Key="Reversal"/>
      <Property Value="true" Key="Cancellation"/>
      <Property Value="false" Key="Loyalty"/>
      <Property Value="false" Key="ManualPAN"/>
    </Core0>
    <Core1 CoreName="oracle.eftlink.opiretail.OPIRetailCore">
      <Property Value="false" Key="Loyalty"/>
      <Property Value="true" Key="Reversal"/>
      <Property Value="false" Key="ManualPAN"/>
      <Property Value="true" Key="Cancellation"/>
```

```
    </Core1>
  </PrivateData>
</ServiceResponse>
```

GetConfig

GetConfig Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceRequest RequestID="3" WorkstationID="Workstation: 10100"
ApplicationSender="EFTLink Load Tester" RequestType="Administration"
RequestSubType="GetConfig">
  <POSdata LanguageCode="en">
    <POSTimeStamp>2021-08-19T14:03:53</POSTimeStamp>
  </POSdata>
</ServiceRequest>
```

GetConfig Response (single active core)

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceResponce RequestID="3" WorkstationID="Workstation: 10100"
ApplicationSender="EFTLink Load Tester" RequestType="Administration"
RequestSubType="GetConfig">
  <PrivateData>
 …
      <Property Value="false" Key="QuickChipEnabled"/>
 …
  </PrivateData>
</ServiceResponse>
```

GetConfig Response (multiple active cores)

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceResponse RequestID="3" WorkstationID="Workstation: 10100"
ApplicationSender="EFTLink Load Tester" RequestType="Administration"
RequestSubType="GetConfig">
  <PrivateData>
 …
      <Property Value="false" Key="QuickChipEnabled"/>
 …
    </Core0>
    <Core1 CoreName="oracle.eftlink.opiretail.OPIRetailCore">
      …
    </Core1>
 </PrivateData>
</ServiceResponse>
```

Initialise

Initialise Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceRequest RequestID="1" WorkstationID="Workstation: 10100"
ApplicationSender="EFTLink Load Tester" RequestType="Administration"
RequestSubType="Initialise">
  <PrivateData>
    <Property Value="20" Key="InvalidCorePromptTimeout"/>
    <Property Value="true" Key="DynamicConfiguration"/>
    <EPSCore Id="0" Name="oracle.eftlink.opiretail.OPIRetailCore">
      <Property Value="false" Key="LineDisplayEnabled"/>
      <Property Value="true" Key="CardAcquisitionEnabled"/>
    </EPSCore>
  </PrivateData>
  <POSdata LanguageCode="en">
    <POSTimeStamp>2021-08-20T13:17:02</POSTimeStamp>
  </POSdata>
</ServiceRequest>
```

Initialise Response

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceResponse RequestID="1" WorkstationID="Workstation: 10100"
ApplicationSender="EFTLink Load Tester" RequestType="Administration"
RequestSubType="Initialise">
  <PrivateData>
    <Property Value="20" Key="InvalidCorePromptTimeout"/>
    <Property Value="true" Key="DynamicConfiguration"/>
    <EPSCore Id="0" Name="oracle.eftlink.opiretail.OPIRetailCore">
      <Property Value="false" Key="LineDisplayEnabled"/>
      <Property Value="true" Key="CardAcquisitionEnabled"/>
    </EPSCore>
  </PrivateData>
  <POSdata LanguageCode="en">
    <POSTimeStamp>2021-08-20T13:17:02</POSTimeStamp>
  </POSdata>
</ServiceResponse>
```

Shutdown

Shutdown Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceRequest RequestID="5" WorkstationID="Workstation: 10100"
ApplicationSender="EFTLink Load Tester" RequestType="Administration"
RequestSubType="Shutdown">
  <POSdata LanguageCode="en">
    <POSTimeStamp>2021-08-20T13:18:01</POSTimeStamp>
  </POSdata>
</ServiceRequest>
```

Shutdown Response

EFTLink doesn't send a response to a shutdown request because it is shutdown.

Uninitialise

Uninitialise Request

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceRequest RequestID="4" WorkstationID="Workstation: 10100"
ApplicationSender="EFTLink Load Tester" RequestType="Administration"
RequestSubType="Uninitialise">
  <POSdata LanguageCode="en">
    <POSTimeStamp>2021-08-20T13:17:23</POSTimeStamp>
  </POSdata>
</ServiceRequest>
```

Uninitialise Response

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceResponse RequestID="4" WorkstationID="Workstation: 10100"
OverallResult="Success" ApplicationSender="EFTLink Load Tester"
RequestType="Administration" RequestSubType="Uninitialise"/>
```

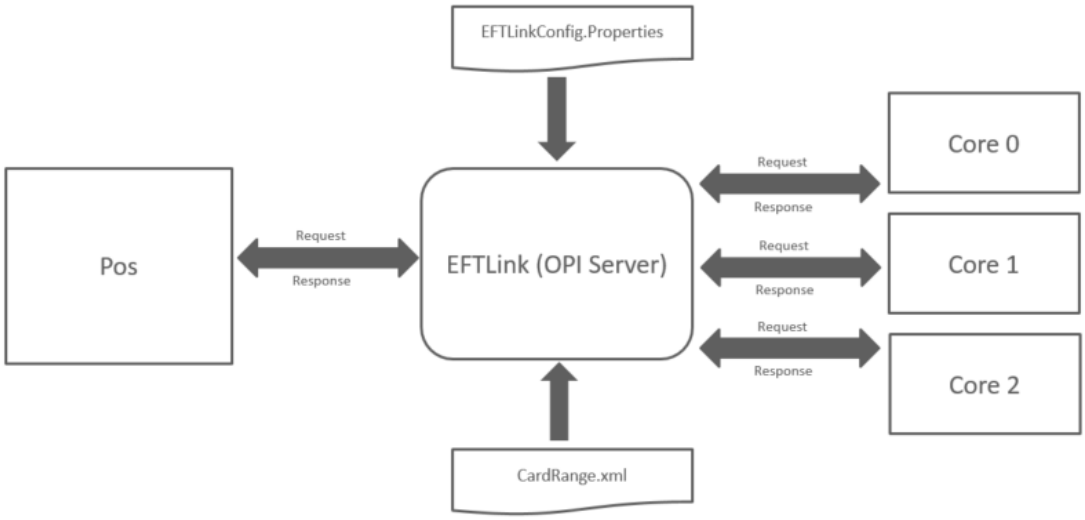
4
Multiple Cores

This chapter describes the concept and procedures of setting up EFTLink using Multiple Cores.

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

- Line Display
- Request Type
- CardRange.xml
- Referrals

Figure 4-1 Multiple Cores



Line Display

Sale State Notification requests can either be disabled altogether or be forwarded to either one 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.

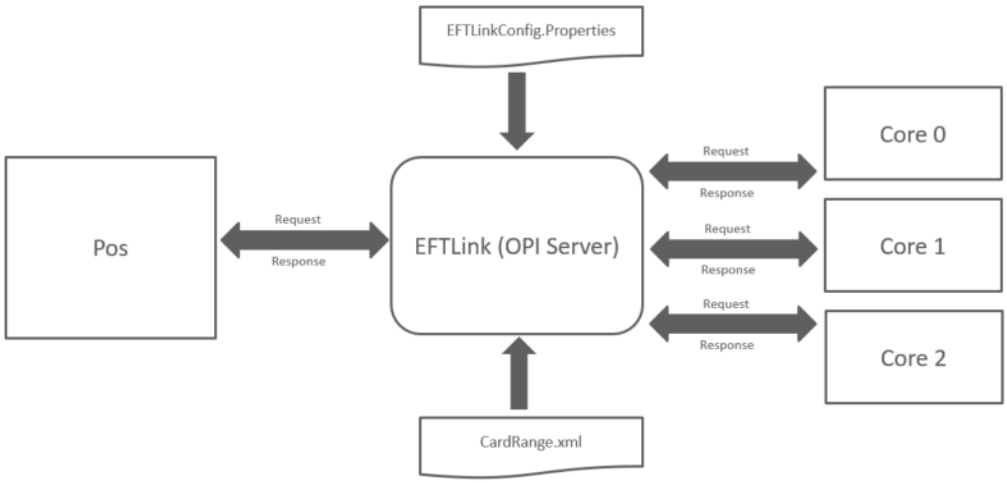
The core requesting the referral is in control and is responsible for the transaction response back to the POS.

EFTLink 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 4-2 Multiple Cores



Line Display

Sale State Notification requests can either be disabled altogether or be forwarded to either one 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.

Multiple Iterations of the Same Core

It is possible to have two or more iterations of a core. For example, if you had two different endpoints that dealt with specific tender types. Ewallet tenders went to one endpoint whereas standard CC/Debit tenders went to a different endpoint.

Such configuration requires two separate `opiretail.properties` file. To accomplish this:

1. Within the `EFTLinkConfig.properties` file Core<x> property, you will need to specify the relative working folder for each core.

   For example:

   EPSCore0 = oracle.eftlink.opiretail.OPIRetailCore WorkingFolder:./OPICore1

   EPSCore1 = oracle.eftlink.opiretail.OPIRetailCore WorkingFolder:./OPICore2

2. In the EFTLink installation folder, create the declared sub folders and place the tailored `opiretail.properties` into their own subfolder.