|  |  |
| --- | --- |
|  | |
| Lightweight Integration LLD | |
| Version: v1.0  Date: 2016.10.03.  Author: Paróczi Zsolt | ENGINEER_klein |

# Modifications

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Modifier** | **Modification** |
| 2016.08.11 | v0.1 | Paróczi Zsolt | Initial version |
| 2016.10.06 | v1.0 | Paróczi Zsolt | 1.0 draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Table of content

[Modifications 2](#_Toc463850671)

[Table of content 3](#_Toc463850672)

[1 Introduction 4](#_Toc463850673)

[1.1 Lightweight integration summary 4](#_Toc463850674)

[1.2 Scope of the document 4](#_Toc463850675)

[1.3 Referenced documents 4](#_Toc463850676)

[1.4 Aim of the document 4](#_Toc463850677)

[1.5 Terminologies and acronyms 4](#_Toc463850678)

[2 Design 6](#_Toc463850679)

[2.1 Architectural design 6](#_Toc463850680)

[2.1.1 System architecture 6](#_Toc463850681)

[2.1.2 Component structure 6](#_Toc463850682)

[2.1.2.1 Proxy 7](#_Toc463850683)

[2.1.3 Core 8](#_Toc463850684)

[2.2 Implementation design 9](#_Toc463850685)

[2.2.1 HaProxy setup 9](#_Toc463850686)

[2.2.1.1 HTTPS termination 10](#_Toc463850687)

[2.2.2 LWI application design 10](#_Toc463850688)

[2.2.2.1 Key classes of the LWI application 10](#_Toc463850689)

[2.2.2.2 LWI handling sequence 10](#_Toc463850690)

[2.2.2.3 LwiHandler 11](#_Toc463850691)

[2.2.2.4 LwiSecurityHandler 12](#_Toc463850692)

[2.2.2.5 LwiRequestBufferingHandler 12](#_Toc463850693)

[2.2.2.6 LwiLogHandler 13](#_Toc463850694)

[2.2.2.7 LwiValidationHandler 13](#_Toc463850695)

[2.2.3 Wildfly/Undertow setup 15](#_Toc463850696)

[2.2.3.1 LWI log setup 15](#_Toc463850697)

[2.2.3.2 LWI service setup 16](#_Toc463850698)

[2.2.3.3 LWI module setup 18](#_Toc463850699)

[2.2.3.4 Generate LWI configuration from YAML files 19](#_Toc463850700)

[3 Appendixes 21](#_Toc463850701)

[3.1 Javadoc 21](#_Toc463850702)

# Introduction

## Lightweight integration summary

There is a need for an alternate lightweight integration solution next to the heavyweight MT OSB standard integration.

The key feature of the new integration solution is the quick development free integration setup.

The new solution is selected for integration in the following circumstances:

* the published service not enterprise service
* API publication is the aim
* the integration is definitely 1:1

The Lightweight Integration solution is based on the HaProxy open source robust proxy application and the Wildfly(JBoss)’s Undertow servlet container. The HaProxy is responsible for the fail over and the load balancing in front of the integration and in the front of the applications. The customized – built in and developed plugins - Undertow module is responsible for the integration requirements.

## Scope of the document

Present document contains the implementation of the Lightweight Integration solution according to the functional specification (HLD).

## Referenced documents

* Lighweight\_Integration\_v7.docx ( requirements )
* Lightweight\_Integration\_HLD\_v1\_1.docx

## Aim of the document

The aim of the present high level document is to describe the implementation of the Lightweight Integration solution according to the high level design document.

## Terminologies and acronyms

|  |  |
| --- | --- |
| **Acronym** | **Description** |
| ESB | Enterprise Service Bus |
| HaProxy | Opensource proxy software solution |
| Lightweight | Simple way |
| LWI | Lightweight Integration |
| OSB | Oracle ESB implementation |
| Undertow | Wildfly(JBoss) servlet container |

# Design

## Architectural design

### System architecture

The lightweight integration architecture looks like the following.



|  |  |  |
| --- | --- | --- |
| **Component** | **Solution** | **Role** |
| Lwi Proxy | Lightweight integration proxy. Based on a HaProxy application. | Responsible for the load balancing and the failover further the https termination and the client certificate attribute as username passing in the header. |
| Lwi Core | Wildfly Undertow customized webserver/servlet container | Responsible for the integrational requirements; log, authentication, authorization, message validation, throttling and routing. |

*Application zone components:*

The several backend applications are accessed from the LightWeight Integration through a proxy layer, therefore the LightWeight Integration has no to take care about the client load balancing.

### Component structure

The LightWeigh integration zone two key components are:

1. Proxy module
2. Customizetd http server module

#### Proxy

The proxy module is based on the high performance and reliable HaProxy load balancer application.

The proxy module responsible for:

* https termination
* load balancing
* Client cert username forward



|  |  |  |
| --- | --- | --- |
| **Component** | **Solution** | **Role** |
| Https termination | SSL setup | For security reason https connection is mandatory from the caller sides. The SSL setup provides the https connection in two manner: one way ( without client cert ) and two way ( with client cert ).  If the ssl is two way keep in mind that the authentication happens here. |
| Load balancing | Load balancer setup | The customized LightWeight Integration module (WildFlyUndertow) can run in several instances. |
| Client cert username forward | Http set header setup | During the https termination the original https encryption cannot be reproduce due to the lack of the client private key, therefore the client cert and its attributes can forwarded in the header for further investigation. This information is used for the authorization. the forwarded attribute is the CN. |

### Core

The core part of the LightWeight integration is the Customized WildFly/Undertow module, this part is responsible for:

* Runtime configuration
* Routing
* Authentication/authorization
* Logging
* Message validation
* Throttling



|  |  |  |
| --- | --- | --- |
| **Component** | **Solution** | **Role** |
| Configuration | Lightweight Integration config XML | This custom XML schema describes the LightWeight integration solution customization possibilities. |
| Configuration | Lightweight Integration config XML transformer | This component transforms the LightWeight Integration configuration into the proper WildFly/Undertow configuration. |
| Configuration | WildFly/Undertow configuration | The core LightWeight Integration solution standard implementation dependent configuration. |
| Routing | Reverse proxy handler | This handler is responsible for forwarding an appropriate incoming request towards the proper host application proxy. |
| Authentication / authorization | Wildfly security module | This component is responsible for the basic authentication and authorization. Authorization means which application has allowed tom call a particular service. |
| Logging | Custom log handler | This handler module is responsible for logging the incoming/outgoing messages in 3 manner: full, context, minimal. |
| Message validation | Custom message validation handler | This handler responsible for validating the incoming messages against its schema. this feature can be turned off. |
| Throttling | Request Limit handler | This handler is responsible for limiting the incoming parallel requests by messages. |

## Implementation design

### HaProxy setup

global

log /dev/log local2

tune.ssl.default-dh-param 2048

defaults

mode http

option httplog

log global

timeout connect 5000ms

timeout client 50000ms

timeout server 50000ms

frontend secure

mode http

# pass the CN in the header

http-request set-header X-SSL-Client-CN %{+Q}[ssl\_c\_s\_dn(cn)]

http-request set-header X-Test-01 Hello

log global

# SSL termination

bind :444 ssl crt /usr/local/etc/haproxy/lwiserver.pem ca-file /usr/local/etc/haproxy/cert/ITEAI2014.ca\_ITEAI2014.crt.pem verify optional

default\_backend wildfly

backend wildfly

balance roundrobin

mode http

server wildfly1 sumatra:446 check

server wildfly2 sumatra:8292 check

frontend stats

bind :445

mode http

stats enable

stats realm Haproxy\ Statistics

stats uri /haproxy\_stats #The URI of the stats page, in this case localhost:445/haproxy\_stats

stats auth ha:ha

#### HTTPS termination

bind :444 ssl crt /usr/local/etc/haproxy/lwiserver.pem ca-file /usr/local/etc/haproxy/cert/ITEAI2014.ca\_ITEAI2014.crt.pem verify optional

verify optional 🡪this means that the 2 way SSL is not forced ( set required for force 2 way )

### LWI application design

The LWI application built on the Wildfly/Undertow pluggable HttpHandler mechanism.

#### Key classes of the LWI application



#### LWI handling sequence

The following sequence diagram shows the sequence of the Lwi handler components filter chain.



#### LwiHandler

|  |  |
| --- | --- |
| **Class name** | hu.telekom.lwi.plugin.LwiHandler |
| **Parameters** | |  |  | | --- | --- | | maxRequests | Maximum parallel requests for a service at once. | | queueSize | Queued requests if the max request reached. | | logLevel | Message log type.  MIN  CTX  FULL | | validationType | Type of the validation.  NO  CTX  MSG | | forceValidation | In case of validation failure if true, it blocks the request otherwise just makes a log entry. | | skipAuthentication | If true the authentication is skipped. | | backEndServiceUrl | URL of the backend service used for the validation. | | backEndConnections | Not used. | | requestTimeout | Not used. | | bufferSize | Size of the inmemory requests n\*16kb | |
| ***handleRequest*** | Main handler class responsible for the proper handler sequence and encapsulate the multiple handler configurations.  order:   * request limit * security * request buffer * log * validation * proxy   This class is the first entry into the LWI process, each request gets an own ID in the LWI and the log will contain this ID for every single entry. This ID is passed by the http exchange object’s attachment. |

#### LwiSecurityHandler

|  |  |
| --- | --- |
| **Class name** | hu.telekom.lwi.plugin. |
| **Parameters** | |  |  | | --- | --- | | skipAuthentication | If true the authentication is skipped. | |
| ***handleRequest*** | The handler is responsible for the authentication and authorization, uses the Undertow internal security framework.  At first it tries to get the basic authentication from the header if not found it looks for the following header key:   * X-SSL-Client-CN   the above key is the user id and means that the client has already authenticated with the HaProxy by client cert.  The basic authentication file is in the Wildfly config dir named:  application-users.properties  If the above two validation fails the handler responds 401.  After the successful authentication, the process validates the acces to the requested service. The service key is from the URL:  /lwi/cnr/getMsisdn 🡪 cnr/getMsisdn  The roles file is stored in the Wildfly config dir named:  application-roles.properties  If the role file has no entry for the user to the requested service the handler responds 401. |

#### LwiRequestBufferingHandler

|  |  |
| --- | --- |
| **Class name** | hu.telekom.lwi.plugin. |
| **Parameters** | |  |  | | --- | --- | | bufferSize | Size of the inmemory requests n\*16kb | |
| ***handleRequest*** | This handler is responsible for intercept the incoming requests if the request size is small enough, less than bufferSize \* 16kb.  If the request is smaller than it cached and passed towards the log and the validator and of course the backend as well.  If the message is larger than the buffer it just pass the fragment the full message validation will skipped but warned in the log. |

#### LwiLogHandler

|  |  |
| --- | --- |
| **Class name** | hu.telekom.lwi.plugin. |
| **Parameters** | |  |  | | --- | --- | | logLevel | Message log type.  MIN  CTX  FULL | |
| ***handleRequest*** | The LwiLogHandler is responsible for the audit log.  MIN  It logs the request time, the caller and the called service  CTX  It logs the request/response time, the caller and the called service with the parsed context information: requested, userId, correlationId  FULL  It logs the request time, the caller and the called service and the complete incoming and outgoing messages.  If tthere is no validation the log reads the response and the request from the channel when that is moved. If The request is validated and the LwiRequestBufferingHandler bufferd the request it logs from the attachment.  request pattern:  [<lwiId>][<timestamp>][<caller> > <called service>][RequestId: <reqid> CorrelationId: <correlid> UserId: <userid>][REQUEST > ....]  response pattern:  [<lwiId>][<timestamp>][<caller> < lwi.estore.cpd][RequestId: <reqid> CorrelationId: <correlid> UserId: <userid>][call: <nn>ms, servicecall: <nn>ms, overhead: <nn>ms][RESPONSE...] |

#### LwiValidationHandler

|  |  |
| --- | --- |
| **Class name** | hu.telekom.lwi.plugin. |
| **Parameters** | |  |  | | --- | --- | | validationType | Type of the validation.  NO  CTX  MSG | | forceValidation | In case of validation failure if true, it blocks the request otherwise just makes a log entry. | | backEndServiceUrl | URL of the backend service used for the validation. | |  |  | |
| ***handleRequest*** | This handler is responsible for the incoming requests validation.  **NO:**  This means that there is no validation at all.  **CTX:**  The CTX validation validates the following context information:   * RequesteId (optional) * CorrelationId (mandatory) * UserId (mandatory)   The path where CTX data is looking for:  TechOSB (XPath):   * //MessageContext/RequestId * //MessageContext/CorrelationId * //MessageContext/UserId   NewOSB (XPath):   * //eiMessageContext/requestId * //eiMessageContext/correlationId * //eiMessageContext/sender   XML attribute:   * RequesteId= * CorrelationId= * UserId =   Header data:   * X-MT-RequestId * X-MT-UserId * X-MT-CorrelationId   **MSG**  The message validation validates the soap request against the wsdl found with the backEndServiceUrl.  The validation process parameters:   * wsdlURL * requestContent   The validation only take place when the full message is in the buffer otherwise a WARN level log happens.  The wsdl validation is only a TEST/DEV feature. |
|  |  |

### Wildfly/Undertow setup

The core layer based on the WildFly’s (JBoss) Undertow http server and servlet container.

The server application can be downloaded from the following url:

<http://wildfly.org/downloads/>

During the implementation we use the Wildfly 10.1 Final version, Servlet-Only Distribution.

*Setup:*

* Extract the downloaded zip into a folder on the target computer.
* Use the standalone.xml configuration and set the following ports properly:
  + management http/https
  + ajp
  + http
  + https (it won’t be used)

*<socket-binding-group name="standard-sockets" default-interface="public" port-offset="${jboss.socket.binding.port-offset:0}">*

*<socket-binding name="management-http" interface="management" port="${jboss.management.http.port:9190}"/>*

*<socket-binding name="management-https" interface="management" port="${jboss.management.https.port:9193}"/>*

*<socket-binding name="ajp" port="${jboss.ajp.port:8109}"/>*

*<socket-binding name="http" port="${jboss.http.port:8192}"/>*

*<socket-binding name="https" port="${jboss.https.port:8193}"/>*

*</socket-binding-group>*

The Wildfly can run in docker as well.

#### LWI log setup

There are 3 type of logs:

1. LWI audit log 🡪 logs the incoming and outgoing messages
2. LWI application log 🡪 logs the LWI process flows
3. Wildfly log 🡪 logs the container processes ( the above (2.) is also included )

##### Loggers

*<logger category="hu.telekom.lwi" use-parent-handlers="true">*

*<level name="DEBUG"/>*

*<handlers>*

*<handler name="LWI\_SEPARATE"/>*

*</handlers>*

*</logger>*

*<logger category="LWI\_LOG\_MESSAGE" use-parent-handlers="false">*

*<level name="DEBUG"/>*

*<handlers>*

*<handler name="LWI\_LOG\_MESSAGE"/>*

*</handlers>*

*</logger>*

##### Log files

*<size-rotating-file-handler name="LWI\_LOG\_MESSAGE" autoflush="true">*

*<level name="INFO"/>*

*<formatter>*

*<pattern-formatter pattern="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n"/>*

*</formatter>*

*<file path="/home/dockeri/logs/lwi\_audit.log"/>*

*<rotate-size value="100M"/>*

*<max-backup-index value="10"/>*

*<append value="true"/>*

*<suffix value=".yyyy-MM-dd"/>*

*</size-rotating-file-handler>*

*<size-rotating-file-handler name="LWI\_SEPARATE" autoflush="true">*

*<level name="INFO"/>*

*<formatter>*

*<pattern-formatter pattern="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n"/>*

*</formatter>*

*<file path="/home/dockeri/logs/lwi\_app.log"/>*

*<rotate-size value="100M"/>*

*<max-backup-index value="10"/>*

*<append value="true"/>*

*<suffix value=".yyyy-MM-dd"/>*

*</size-rotating-file-handler>*

#### LWI service setup

The LWI is set up by the standard Wildfly/Undertow handler configurations in the standalone.xml.

There are 3 configured handler:

1. Custom LwiHandler 🡪 this wraps the LWI custom handlers
2. Reverse Proxy Handler 🡪 this forwards the request to the business backends
3. LWI request limit handler 🡪 limits the request in the LWI regardless of the message type

##### Lwi custom handler setup

*<!—section undertow -->*

*<subsystem xmlns="urn:jboss:domain:undertow:3.1">*

*...*

*<!—section server*

*<server name="default-server">*

*...*

*<!—section host -->*

*<host name="default-host" alias="localhost">*

*...*

*<!-- CNR getMsisdn -->*

*<filter-ref name="lwi-cnr-getMsisdn" predicate="path-prefix('/lwi/cnr/getMsisdn')"/>*

*...*

*</host>*

*</server>*

*...*

*<!-- section filters -->*

*<filters>*

*<!-- CNR getMsisdn -->*

*<filter name="lwi-cnr-getMsisdn" class-name="hu.telekom.lwi.plugin.LwiHandler"*

*module="hu.telekom.lwi">*

*<!-- maximum number of the parallel request for the service -->*

*<param name="maxRequests" value="10"/>*

*<!-- queue length for the service -->*

*<param name="queueSize" value="1"/>*

*<!-- audit log level [MIN|CTX|FULL] -->*

*<param name="logLevel" value="FULL"/>*

*<!-- type of the validation [MSG|CTX|NO] -->*

*<param name="validationType" value="CTX"/>*

*<!-- this can switch the validation off -->*

*<param name="skipAuthentication" value="false"/>*

*<!-- url for the full msg validation validationType=MSG to get the wsdl -->*

*<param name="backEndServiceUrl"*

*value="http://10.20.233.130:8091/LwiMockTargets/GetMsisdn"/>*

*<!-- not used -->*

*<param name="backEndConnections" value="1"/>*

*<!-- not used -->*

*<param name="requestTimeout" value="20000"/>*

*<!-- size of the maximum buffered message n\*16kB -->*

*<param name="bufferSize" value="20"/>*

*</filter>*

*</filters>*

*...*

*</subsystem>*

##### Reverse proxy handler setup

*<!-- section management -->*

*<management>*

*<security-realms>*

*...*

*<security-realm name="UndertowRealm">*

*<server-identities>*

*<ssl>*

*<keystore path="wildfly-rsa.jks"*

*relative-to="jboss.server.config.dir"*

*keystore-password="123456" alias="wildfly"*

*key-password="123456"/>*

*</ssl>*

*</server-identities>*

*<authentication>*

*<truststore path="wildfly-rsa.jks"*

*relative-to="jboss.server.config.dir"*

*keystore-password="123456"/>*

*</authentication>*

*</security-realm>*

*</security-realms>*

*...*

*</management>*

*<!-- section undertow -->*

*<subsystem xmlns="urn:jboss:domain:undertow:3.1">*

*...*

*<!—section server*

*<server name="default-server">*

*...*

*<!—section host -->*

*<host name="default-host" alias="localhost">*

*...*

*<!-- CNR getMsisdn -->*

*<location handler="lwi-cnr-getMsisdn-proxy" name="/lwi/cnr/getMsisdn" />*

*...*

*</host>*

*</server>*

*...*

*<!— section handlers -->*

*<handlers>*

*<reverse-proxy name="lwi-cnr-getMsisdn-proxy"*

*<!-- request timout -->*

*max-request-time="30000"*

*<!-- number of the connections to the backend/thread -->*

*connections-per-thread="1">*

*<host name="cnr-server1"*

*<!-- https -->*

*scheme="https"*

*<!-- wildfly realm for ssl timout -->*

*security-realm="UndertowRealm"*

*<!—outbound host name see below -->*

*outbound-socket-binding="cnr-server"*

*<!-- backend service URL -->*

*path="/LwiMockTargets/GetMsisdn"*

*instance-id="cnr-server-route1"/>*

*</reverse-proxy>*

*</handlers>*

*...*

*</subsystem>*

*...*

*<!-- section socket bindings -->*

*<socket-binding-group name="standard-sockets" default-interface="public" port-offset="${jboss.socket.binding.port-offset:0}">*

*...*

*<!-- CNR server -->*

*<outbound-socket-binding name="cnr-server" >*

*<remote-destination host="localhost" port="8095"/>*

*</outbound-socket-binding>*

*</socket-binding-group>*

##### LWI request limit handler

*<!—section undertow -->*

*<subsystem xmlns="urn:jboss:domain:undertow:3.1">*

*...*

*<!—section server*

*<server name="default-server">*

*...*

*<!—section host -->*

*<host name="default-host" alias="localhost">*

*...*

*<!-- LWI -->*

*<filter-ref name="FULL\_REQUEST\_LIMIT" predicate="path-prefix('/lwi')"/>*

*...*

*</host>*

*</server>*

*...*

*<!-- section filters -->*

*<filters>*

*<request-limit name="FULL\_REQUEST\_LIMIT"*

*max-concurrent-requests="10"*

*queue-size="10"/>*

*...*

*</filters>*

*...*

*</subsystem>*

#### LWI module setup

The LWI module – build by Maven - is installed as a Wildfly module.

*component name:*

* hu.telekom.lwi

*module path:*

* wildfly-servlet-10.0.0.Final\modules\hu\telekom\lwi\main\

*module dir:*

* module.xml
* lwi-wildflyplugin-<version>.jar

*module.xml*

*<?xml version="1.0" encoding="UTF-8"?>*

*<module xmlns="urn:jboss:module:1.3" name="hu.telekom.lwi">*

*<resources>*

*<resource-root path="wfplugins-1.0.0.jar"/>*

*</resources>*

*<dependencies>*

*<module name="io.undertow.core"/>*

*<module name="org.jboss.xnio"/>*

*<module name="org.jboss.logging"/>*

*</dependencies>*

*</module>*

#### Generate LWI configuration from YAML files

For keeping the configuration as simple as possible the whole LWI has own configuration structure based on 2 + n pieces yaml files.

The configuration files are stored in the git config folder of the LWI separated by environments

##### Server configuration yml

*# lwi server section*

*logger:*

*- name: messageLog*

*logfile: mlog.log*

*rollsize: 10M*

*rollbackup: 20*

*- name: appLog*

*logfile: mlog.log*

*rollbackup: 20*

*rollsize: 10M*

*load:*

*- maxRequests: 100*

*- queueSize: 100*

##### Security configuration yml

*# lwi consumers*

*consumers:*

*- name: test1*

*password: qwertz*

*- name: test2*

*password: qwertz*

*- name: test3*

*password: qwertz*

##### n pieces service configuration

*service:*

*- name: lwi-estore-qpdl*

*maxRequests: 20*

*queueLength: 10*

*validationType: CTX*

*logLevel: FULL*

*skipAuthentication: false*

*backEndServiceUrl: http://localhost:8091/LwiMockTargets/Qpdl*

*backEndConnections: 10*

*requestTimeout: 10000*

*bufferSize: 10*

*consumers: [test1,test2,test3]*

##### Yml to standalone.xml converter

Script name: lwi\_create\_config.pl

*parameterization:*

* userDat </path/to/file> (users.yml file path)
* serverDat </path/to/file> (server.yml path)
* appUsers </path/to/file> (output application-users.properties path)
* appRoles </path/to/file> (output application-roles.properties path)
* serviceDatMask <fileMask> (mask of the service.yml files even with full path. „\*service.yml” The \* char should be protected from the shell)

The user and role files are the standard Wildfly configurations but the service/server generation result is a jboss.cli file that should run against a wildfly installation to configure the server.

# Appendixes