|  |  |
| --- | --- |
|  | |
| Lightweight Integration HLD | |
| Version: v1.2  Date: 2016.09.18.  Author: Paróczi Zsolt | ENGINEER_klein |

# Modifications

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Modifier** | **Modification** |
| 2016.07.03 | v1.0 | Paróczi Zsolt | Initial version |
| 2016.09.06 | v1.1 | Paróczi Zsolt | Some fix |
| 2016.09.18 | v1.2 | Paróczi Zsolt | Modification during the implementation phase |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Table of content

[Modifications 2](#_Toc456096827)

[Table of content 3](#_Toc456096828)

[1 Introduction 4](#_Toc456096829)

[1.1 Lightweight integration summary 4](#_Toc456096830)

[1.2 Scope of the document 4](#_Toc456096831)

[1.3 Referenced documents 4](#_Toc456096832)

[1.4 Aim of the document 4](#_Toc456096833)

[1.5 Terminologies and acronyms 4](#_Toc456096834)

[2 High level requirements 6](#_Toc456096835)

[3 Design 7](#_Toc456096836)

[3.1 Architectural design 7](#_Toc456096837)

[3.1.1 System architecture 7](#_Toc456096838)

[3.1.2 Component structure 7](#_Toc456096839)

[3.1.2.1 Proxy 8](#_Toc456096840)

[3.1.3 Core 9](#_Toc456096841)

[3.2 Functional design 10](#_Toc456096842)

[3.2.1 Proxy layer 10](#_Toc456096843)

[3.2.1.1 HTTPS termination 10](#_Toc456096844)

[3.2.1.2 Load balancing 11](#_Toc456096845)

[3.2.1.3 Client username forward from client cert 11](#_Toc456096846)

[3.2.2 Core layer 11](#_Toc456096847)

[3.2.2.1 Configuration 12](#_Toc456096848)

[3.2.2.2 Routig 13](#_Toc456096849)

[3.2.2.3 Authentication and authorization 13](#_Toc456096850)

[3.2.2.4 Message logging 15](#_Toc456096851)

[3.2.2.5 Message validation 16](#_Toc456096852)

[3.2.2.6 Message throttling 16](#_Toc456096853)

[3.2.2.7 xrt 16](#_Toc456096854)

[4 Appendixes 17](#_Toc456096855)

[4.1 Security configuration schema 17](#_Toc456096856)

[4.2 Configuration schema 17](#_Toc456096857)

[4.3 Configuration sample 17](#_Toc456096858)

# 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 functionality of the Lightweight Integration solution according to the requirements.

## Referenced documents

* Lighweight\_Integration\_v7.docx ( specification )

## Aim of the document

The aim of the present high level document is to describe the functionality of the Lightweight Integration solution according to the requirements defined in the specification documents.

## Terminologies and acronyms

|  |  |
| --- | --- |
| **Acronym** | **Description** |
| ESB | Enterprise Service Bus |
| Haldler | Undertow filter like component to handle an incoming request in custom way. |
| Lightweight | Simple way |
| OSB | Oracle ESB implementation |
| Undertow | Wildfly(JBoss) servlet container |
|  |  |

# High level requirements

This section details the high level requirements identified according to the IAD documents.

|  |  |
| --- | --- |
| **ID** | **Description** |
| REQ-LWI-001 | Proxy + customized java based http server. |
| REQ-LWI-002 | Routing the incoming request to the provider service from the central lightweight integration address. |
| REQ-LWI-003 | Audit log. Log the incoming and outgoing messages. The log detail can be switched by the followings; full, context, minimal (date, caller, service name). |
| REQ-LWI-004 | Secure connection over https, optionally in case of client certificate the cert details should be used as user authentication. |
| REQ-LWI-005 | Authentication and authorization.  Basic authentication and client certificate authentication support.  Authorization: caller/operation 🡪 only the allowed services can be called by a consumer |
| REQ-LWI-006 | Optional message validation, full soap message validation, message standard validation ( techOSB, newOSB standard format, message attributes, http header values ). |
| REQ-LWI-007 | Message throttling, limiting the maximum number of requests by operation. |
| REQ-LWI-008 | Runtime deployment and configuration (without server restart). |
| REQ-LWI-009 | Low latency and high performance, minimal memory usage. |
|  |  |

# 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. |

## Functional design

### Proxy layer

The proxy layer is based on the HaProxy high performance reliable open source proxy solution.

#### HTTPS termination

**Business requirement reference**

* *REQ-LWI-004*

The SSL termination happens on the HaProxy layer. The HaProxy natively supports the SSL.

*frontend localhost*

*bind \*:80*

*bind \*:443 ssl crt server.pem ca-file chain.pem verify optional*

*redirect scheme https if !{ ssl\_fc }*

*mode http*

*default\_backend nodes*

The above setup redirects the http to https also and sets the server cert for the ssl, moreover tells the caller to provide the client cert if it is available.

#### Load balancing

**Business requirement reference**

* *REQ-LWI-001*

The HaProxy behaves as load balancer in the front of the core LightWeight Integration

*backend nodes*

*mode http*

*balance roundrobin*

*option forwardfor*

*server lightint01 192.168.0.1:7211 check*

*server lightint02 192.168.0.1:7211 check*

#### Client username forward from client cert

**Business requirement reference**

* *REQ-LWI-004*

If the client provides a certificate, the HaProxy SSL setup ( CA configuration ) will identify the client itself therefore the username is needed in the core layer to authorize the requested service for the requestor client. The HaProxy has the ability to forward the client cert attributes in the request.

*frontend localhost*

*.*

*.*

*http-request set-header X-SSL-Client-DN %{+Q}[ssl\_c\_s\_dn]*

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

*http-request set-header X-SSL-Issuer %{+Q}[ssl\_c\_i\_dn]*

*.*

*.*

The following attributes are forwarded:

* Common name CN
* Distinguished name DN
* Certificate Issuer

### Core layer

The core layer based on the WildFly’s (JBoss) Undertow http server and servlet container. The core layer has two main parts:

1. Configuration
2. Customized http server

#### Configuration

**Business requirement reference**

* *REQ-LWI-008*

The core component customized Wildfly server module’s configuration is a separate independent module with two parts:

|  |  |
| --- | --- |
| 1. *configuration schema* | The schema describes the LightWeight Integration module configuration. |
| 1. *configuration transformer* | The transformer transforms the LightWeight Integration module configuration into the WildFly standard configuration. |

##### Configuration schemas

The LWI module has 3 types of configuarations:

|  |  |
| --- | --- |
| 1. *Consumers configuration* | Technically this is a simple user database who has access to the LWI. the passwords are stored here as well. |
| 1. *LWI server configuration* | This configuration relates to the Wildfly server instance that runs the LWI. The configuration covers the log setup and the throttling of the LWI server instance. |
| 1. *Service configuration* | Service related configuration that handled by the LWI. |

The configuration files are in a human readable YAML file format.

|  |  |
| --- | --- |
| ***Consumer configuration*** | |
| ***attribute*** | **description** |
| username | Application username. |
| password | Application password. |

|  |  |
| --- | --- |
| ***LWI server configuration*** | |
| ***attribute*** | **description** |
| Max allowed requests | Maximum number of the parallel allowed request. |
| Request queue size | Queue size if the max allowed request number reached. |
| application log file | Separate LWI application log file path. |
| application log size | Separate LWI application log size. |
| application roll count | Separate LWI application rolled log file count. |
| message log file | Separate LWI message log file path. |
| message log size | Separate LWI message log size. |
| message roll count | Separate LWI message rolled log file count. |

|  |  |
| --- | --- |
| ***LWI server configuration*** | |
| ***attribute*** | **description** |
| Service name | Name of the service |
| Service URI | URI of the service published by the LWI:  lwi/provider/service > lwi/cnr/getMsisdn |
| Max allowed requests | Maximum number of the parallel allowed request for this service |
| Request queue size | Queue size if the max allowed request number reached on the service. |
| Log level | How the LWI logs a service, full message, only the context, minimal service call. |
| Validation level | How the LWI validates an incoming message, full soap request, only the required context attributes or nothing.  MSG|CTX|NO |
| Backend service URL | URL of the backend service that is published on the LWI. |
| Skip authentication | For testing purpose the authentication can be switched off. |
| Request timeout. | Timeout of the request. |
| Allowed consumers | List of users ( consumers ) who can call the service. |



##### Configuration transformer

The transformer application is a command line tool that transforms the LWI configurations into the Wildfly appropriate configurations ( standalone.xml, application-users.properties, application-roles.properties ).

#### Routig

**Business requirement reference**

* *REQ-LWI-002*

The key part of the integration is the routing. All the integrated applications entry point is a single host (clustered) to provide control over the interactions between the individual applications.

Each service has an URI and a back end URL in the LWI configuration.

publish URI 🡪 the LWI publish the service on this path.

backend URL 🡪 the LWI forwards the requests to this path.

#### Authentication and authorization

**Business requirement reference**

* *REQ-LWI-005*

The authentication can happen the following ways:

1. basic authentication
2. client cert authentication

The authentication and the authorization is served by a custom handler.

|  |  |
| --- | --- |
| ***LwiSecurityHandler*** | |
| **attribute** | **description** |
| securityConfigFile | Path to the security xml file. |
| certUserNameAttribute | In case of cert based authentication the header attribute name the contains the user name.  e.g.: X-MT-LWI-CERTUSERNAME |

*Authentication, authorization process flow:*



|  |  |
| --- | --- |
| *Checks username from cert* | This step gets the http request header and cheks wether the HaProxy layer has passed the username token from the cert. The attribute name is configured in the *certUserNameAttribute* of the security handler. |
| *Check username/pwd from config* | This step gets the username and the password from the http header basic authentication attribute ( base64 encoded ) and validate against the configuration file; find the appropriate caller by the name and validates the password. |
| *Check allowed operations* | If the caller authenticated the authorization step checks the called operation whether is accessible by the caller.  Operation name: <provider>/<operation> 🡪 cnr/getmsisdn |
| *Authentication error* | This step breaks the caller chain and responds http 401. |
| *Authorization error* | This step breaks the caller chain and responds http 403. |

#### Message logging

**Business requirement reference**

* *REQ-LWI-003*

The Lightweight integration component can log the incoming requests and responses. The message logging is provided by a custom handler.

The handler uses the JBoss standard logger.

|  |  |
| --- | --- |
| ***LwiMessageLogHandler*** | |
| **attribute** | **description** |
| level | Detail of the log: FULL|CTX|MIN  FULL > total incoming request without attachment and the headers  CTX > message context  MIN > caller,provider,operation,date |

*Log format:*

[Caller>Provider.Operation][ context ][ message ]

The message is flattened there are no line breaks in the log within one message.

*Context data:*

The followings are in the context part if it can be identified.

|  |  |  |
| --- | --- | --- |
| **attribute name** | **required** | **description** |
| RequestId | yes | Unique message ID |
| CorrelationId | no | Unique of the e2e business process |
| UserId | no | User id of the requestor if any |

If the context data cannot be identified, the context part contains the N/A constant.

*Parsing order for resolving the message attributes:*

1. Soap message attribute  
   The required attributes can arrive in the first soap element under the soap body.  
   *<soapenv:Body>  
    <CreateProducedDocument RequestId=”1234566” CorrelationId=”21212”* UserId=”TOTHA”
2. According to the TechOSB MessageContext complex type
3. According to the NewOSB eiMessageContext
4. Http header  
   X-MT-RequestId=1234566  
   X-MT-CorrelationId=21212  
   X-MT-UserId=”TOTHA”

#### Message validation

**Business requirement reference**

* *REQ-LWI-006*

The Lightweight integration component can validate the incoming requests. The validation is provided by a custom handler.

|  |  |
| --- | --- |
| ***LwiMessageValidationHandler*** | |
| **attribute** | **description** |
| validationTpye | Types: CTX|MSG|NO  CTX > message context  MSG > soap message validation  NO > no validation |
| wsdlLocation | URL of the wsdl – online provided service WSLD |

*Validation process flow:*



|  |  |
| --- | --- |
| *Validate context* | *Context validation order:*   1. Soap message attribute 2. According to the TechOSB MessageContext complex type 3. According to the NewOSB eiMessageContext 4. Http header   *See the previous chapter for more detail.* |
| *Validate soap message* | The soap message validate against the service provider application wsdl. the wsdl url is part of the handler configuration. |
| *Validation error* | This step breaks the caller chain and responds http 500. |

#### Message throttling

**Business requirement reference**

* *REQ-LWI-007*

Each operation can be limited by the number of the parallel calls. The limitation has two attributes, maximum parallel request number and the queue size.

This requirement fulfilled at low level by the Undertow built in RequestLimitingHadler.

# Appendixes

## Consumer (user) configuration

*# lwi consumers*

*consumers:*

*- name: test1*

*password: qwertz*

*- name: test2*

*password: qwertz*

*- name: test3*

*password: qwertz*

## Server configuration

*# 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*

## 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*

*consumers: [test1,test2,test3]*