Table of contents

1 About USEF 2

1.1 About the Reference Implementation 2

1.2 References 2

1.3 Obtaining the source code 2

2 Using USEF 3

2.1 Environment variables 3

3 Prerequisites 4

3.1 Oracle Java SE Development Kit 8 or OpenJDK JDK 8: 4

3.2 Apache Maven 4

3.3 H2 4

3.4 Libsodium 5

3.5 Root certificate (optional, for demo purposes). 5

3.6 ISC BIND 6

4 Starting and stopping the USEF environment 8

4.1 Starting the USEF environment 8

4.2 Stopping the USEF environment 8

4.3 Accessing the USEF database 8

4.4 Sending messages 10

4.5 JBoss Management Console 10

4.6 Nodes folder 11

5 Configuration 12

5.1 Adjusting the usef-environment.yaml configuration 12

5.2 Logging 12

5.3 Setting up the database with Common Reference information 13

5.4 Reserved TCP ports 13

5.5 Configure resolver entries 14

5.6 Configuring a proxy server (optional) 14

6 Resolving participant information 16

6.1 Secure information provision using DNSSEC 16

6.2 By default, the configuration of Bind as described earlier in ‘ISC BIND 16

6.3 Configuring DNS in wildfly 16

6.4 Resolving without DNS server 16

7 Appendix – Configuration of scheduled processes 17

7.1 Extraday triggers 17

7.2 Intraday triggers 18

# About USEF

The Universal Smart Energy Framework (USEF) developed by the USEF Foundation provides non-discriminatory access to smart energy systems at acceptable cost-to-connect and cost-to-serve levels.

By providing an open and consistent framework of specifications, designs, and implementation guidelines [2], USEF enables participants to seamlessly co-create a fully functional smart energy system.

The USEF Foundation acts as the framework’s steward and aspires to establish it as the de facto framework for smart energy products, services, and solutions.

In 2020 the foundation wants to be part of 25% of all smart energy systems in at least 5 different markets throughout Europe—and, hopefully, beyond.

## About the Reference Implementation

The USEF Reference Implementation is made available in the form of downloadable source code. It is the definitive interpretation of the USEF 2015 specification [3] and provides a starting point for third parties aiming to commercially exploit (parts of) the design or aiming to build products and services using an implementation of the design.

## References

|  |  |
| --- | --- |
| **Reference** | **Description** |
| [1] | USEF The Framework Implemented – System Architecture.doc |
| [2] | USEF The Framework Implemented – Implementation Guidelines.doc |
| [3] | USEF Specifications 2015 |

## Obtaining the source code

The source code can be obtained from the USEF Foundation’s GitHub repository using the following link:

https://github.com/USEF-Foundation/ri.usef.energy

If you want to get involved or would like further information, please see:

http://www.usef.energy/Get-involved.aspx

# Using USEF

The source code deliverables consist of several folders:

* + - usef-build: Which contains the USEF source code to build all participant components.
    - usef-environment: Which contains scripts and configuration to run USEF.
    - usef-environment-tool: Which contains source code to build tooling to generate the USEF environment.

When starting development, the folder usef-build is important and contains pom-files to build the source code with Maven.

When you start using USEF, the folder usef-environment is important. This deliverable contains several folders:

* + - bin: Which contains Windows batch scripts and Linux scripts the generate the environment and start the environment.
    - config: Which contains configuration (which can be adjusted). The file usef-environment.yaml is used to define the configuration of the environment. The environment contains configuration of all partipants.
    - lib: Do not change the content of this folder! At the moment it contains a library for encryption.
    - template: This folder contains template files which can be changed by advanced users. The templates are used when the USEF environment is generated.

## Environment variables

The following environment variables need to be defined correctly to be able to run the scripts which are covered in this document:

* + - JAVA\_HOME – needs to be set to the location of the folder in which the Java JDK 8 will be installed.
    - PATH – must contain a reference to the location of the bin folder in the folder where Apache Maven has been installed, and it must contain the directory $JAVA\_HOME/bin (or %JAVA\_HOME%\bin on Windows). If BIND is used on the same Windows machine, the path “C:\Program Files\ISC BIND 9\bin” must be added too.
    - LD\_LIBRARY\_PATH – (UNIX-only) must contain the path “/usr/local/lib”, as that’s the recommended place where libsodium will be installed.
    - JBOSS\_HOME – needs to be set to the location of the folder in which JBoss Wildfly will be installed.
    - USEF\_HOME – needs to be set to the location of the folder where the repository was cloned.

# Prerequisites

For building and using USEF, you’ll need the following tooling:

## Oracle Java SE Development Kit 8 or OpenJDK JDK 8:

To build and run source code, you need at least one of these Java 8 JDK’s. This can be downloaded from

<http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html#javasejdk>

Or follow the instructions from

<http://openjdk.java.net/install/>

Note 1: You explicitly have to download and install the JDK. Installing just the JRE (Java Runtime Environment) is not sufficient for building a USEF environment.  
Note 2: Use the 64 bit version of the JDK, because we need the 64-bit encryption libraries.

## Apache Maven

To build and run source code, you need at least Apache Maven version 3.2.3. The latest stable version can be downloaded from:

<http://maven.apache.org/download.cgi>

Note that the environment variables (M2\_HOME, M2) mentioned in the Maven installation instructions are not necessary for the USEF environment.

JBoss Wildfly: To run USEF on an application server, JBoss Wildfly version 10.0.0.Final is needed, which can be downloaded from:

<http://wildfly.org/downloads>

Please note that the USEF development team has only tested with version 10.0.0.Final. We strongly recommend that only this version of JBoss Wildfly is used, and not later versions.

## H2

The USEF Reference Implementation is shipped with the H2 database for convenience purposes only. This database should not be used in production environments or any other USEF application where high performance and data integrity are deemed critical.

JBoss Wildfly 10.0.0 Final is shipped with the H2 Database Engine version 1.3.173. This version contains bugs that prevents the USEF Reference Implementation from working correctly. Beta version 1.4.190 does not contain these bugs.

Follow these instruction to downgrade:

* + - Download the Platform Independent Zip for H2 1.4.190 from  
        
      <http://www.h2database.com/html/download.html>
    - Unzip the downloaded file
    - Copy the file h2/bin/h2-1.4.190.jar into ${JBOSS\_HOME}/modules/system/layers/base/com/h2database/h2/main
    - Remove h2-1.3.173.jar from ${JBOSS\_HOME}/modules/system/layers/base/com/h2database/h2/main
    - Modify the resource root source path in ${JBOSS\_HOME}/modules/system/layers/base/com/h2database/h2/main/module.xml into h2-1.4.190.jar.

## Libsodium

To be able to securely transmit and authenticate messages, Libsodium 1.0.10 must be built and installed.

On Windows:

* + - Download <https://download.libsodium.org/libsodium/releases/libsodium-1.0.10-mingw.tar.gz> to a new directory and unpack it with your favorite tool (e.g. 7-zip).
    - Copy the file “libsodium-win64/bin/libsodium-18.dll” to some directory in your PATH, e.g. %JAVA\_HOME%/bin, and rename the file to “sodium.dll.

On other platforms:

* + - Make sure you have gcc and make in your environment:  
       sudo apt-get install gcc make
    - Download <https://download.libsodium.org/libsodium/releases/libsodium-1.0.10.tar.gz> to a new empty directory and start in this directory:   
       cd <build\_directory>
    - Unpack the archive, build it and install it:  
       tar xfz libsodium-1.0.10.tar.gz  
       cd libsodium-1.0.10  
       ./configure && make  
       sudo make install
    - Add /usr/local/lib to the LD\_LIBRARY\_PATH by adding the following to :  
      export LD\_LIBRARY\_PATH=/usr/local/lib:$LD\_LIBRARY\_PATH

## Root certificate (optional, for demo purposes).

The reference implementation contains a Java key store with 3 entries, for demo purposes: usef\_ri/usef-enviroment/template/usef.jks.   
  
You can extract the root CA certificate from this file as follows:  
 cd <path\_to>/usef\_ri/usef-environment/template  
 keytool -exportcert -keystore usef.jks -alias usef\_ri -file rootCA.der  
 (password: “usef1234”)

This certificate needs to be installed in the Oracle JVM key store.

Do the following with administrator rights:

cd $JAVA\_HOME/jre/lib/security  
 keytool -importcert -keystore cacerts –file \  
 <path\_to>/usef\_ri/usef-environment/template/rootCA.der (password: “changeit”)

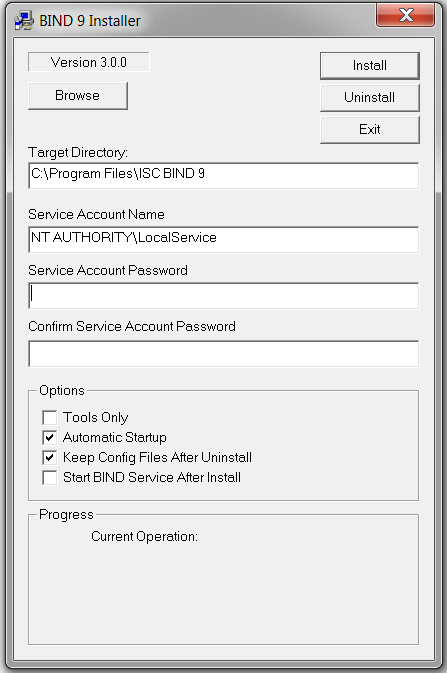
If you use another tool to access USEF using https, and this tool uses its own key store (e.g. postman), you will need to install this root CA certificate file there too.

## ISC BIND

Installing ISC BIND 9.10.x is optional in demo environment, mandatory in production.

The procedure as described here is intended as a simple example and works for the default USEF configuration. For more information, please consult the Bind 9 Administrator Reference Manual[[1]](#footnote-1).

On windows:

* + - Download <ftp://ftp.isc.org/isc/bind9/9.10.2/BIND9.10.2.x64.zip> (or later) to a new directory and unpack it.
    - Locate the file “BINDInstall.exe”, right-click on it, select “Run as administrator”. You will see the following dialog:  
      
    - Enter the information as shown, the two password fields can be left empty. Then press “Install”.
    - Copy the files “named.conf” and “usef\_bind.zone” to your “C:\Program Files\ICS BIND 9\etc” directory.   
      You can find those files in “usef\_ri/usef-environment/config”.
    - Manually change the following line in named.conf (as administrator):  
       directory "/var/cache/bind";  
      to:  
       directory "C:\Program Files\ISC BIND 9\etc";
    - Be sure that the "C:\Program Files\ISC BIND 9\bin" folder has been added to the PATH as described in Environment variables.
    - Add Full Control Rights for user “LOCAL SERVICE” to folder "C:\Program Files\ISC BIND 9\etc".
    - Start the "ISC BIND" Service.

On other platforms (e.g. Ubuntu 14.04):

* + - Excute:  
       sudo apt-get install bind9
    - Copy the file usef\_ri/usef-environment/config/named.conf to your /etc/bind directory.
    - Copy the file usef\_ri/usef-environment/config/usef\_bind.zone to your /var/cache/bind directory.
    - Restart bind by executing:  
       sudo service bind9 restart

# Starting and stopping the USEF environment

The USEF environment can be started on Windows or Linux. The usef-environment/bin folder contains Windows batch scripts and Linux scripts:

* + - cleanup.cmd and cleanup.sh: Clean the USEF environment and stops the H2 database.
    - prepare.cmd and prepare.sh: Build USEF and generates the USEF environment before it can be started.
    - start-h2-database.cmd and start-h2-database.sh: Start the H2 database as a separate process.
    - start-usef-environment.cmd and start-usef-environment.sh: Start the USEF environment with JBoss Wildfly.
    - stop-h2-database.cmd and stop-h2-database.sh: Stops the H2 database.
    - stop-usef-environment.cmd and stop-usef-environment.sh: Stop the USEF environment.

When you start using USEF, run the scripts in the order as mentioned to get the USEF environment up and running. When you do this the first time, you do not have to run the cleanup script.

## Starting the USEF environment

To start the USEF environment:

* + - Run the prepare script (The parameter –skipBuild skips building USEF, when USEF has already been built)
    - Run the start-h2-database script.
    - Run the start-usef-environment script
    - After the start-usef-environment script is executed, you can start sending messages to the participants.

## Stopping the USEF environment

To stop the USEF environment:

* + - Stop the USEF Environment manually by running the stop-usef-environment script
    - Stop the USEF database by running the stop-h2-database script
    - Run the cleanup script

## Accessing the USEF database

All participants, which are configured in the usef-environment.yaml file, are located in 1 H2 database file which is encrypted by AES.

To access the USEF database, you can use your favorite web-browser to connect to the following location

<http://localhost:8082/>

At connection, the following screen appears:

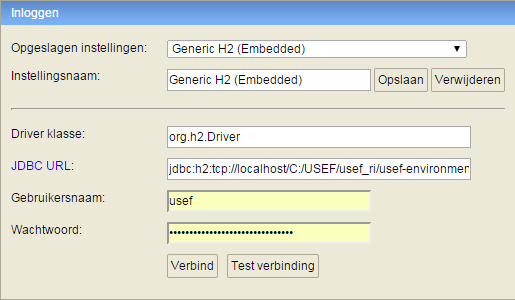


Figure 1: H2 Database connection dialog

In this dialog, the Driver, JDBC URL, username and password have to be filled according to the contents of the following file:

usef-environment/nodes/localhost/configuration/credentials.properties

The username is ‘usef’ without quotes and the Driver must be org.h2.Driver. The password and JDBC URL can be extracted from the DB\_PASSWORD and DB\_URL fields from the credentials.properties (excluding the backslashes).

Note: this file must be removed from the file system to be USEF compliant.

## Sending messages

By default, 8 different participants are deployed which can be accessed through their own URL:  
 AGR1: <https://agr1.usef-example.com:8443/agr1.usef-example.com_AGR/rest/MessageService/sendMessage>  
 AGR2: <https://agr2.usef-example.com:8443/agr2.usef-example.com_AGR/rest/MessageService/sendMessage>  
 AGR3: <https://agr3.usef-example.com:8443/agr3.usef-example.com_AGR/rest/MessageService/sendMessage>  
 BRP1: <https://brp1.usef-example.com:8443/brp1.usef-example.com_BRP/rest/MessageService/sendMessage>  
 BRP2: <https://brp2.usef-example.com:8443/brp2.usef-example.com_BRP/rest/MessageService/sendMessage>  
 CRO1: <https://cro1.usef-example.com:8443/cro1.usef-example.com_CRO/rest/MessageService/sendMessage>  
 DSO1: <https://dso1.usef-example.com:8443/dso1.usef-example.com_DSO/rest/MessageService/sendMessage>  
 MDC1: <https://mdc1.usef-example.com:8443/mdc.usef-example.com_MDC/rest/MessageService/sendMessage>

The name of the domain (like agr1.usef-example.com) is configured in the usef-environment.yaml file.

With HTTP Post (header should be Content-Type = text/xml), an XML message can be send to one of the mentioned URLs:

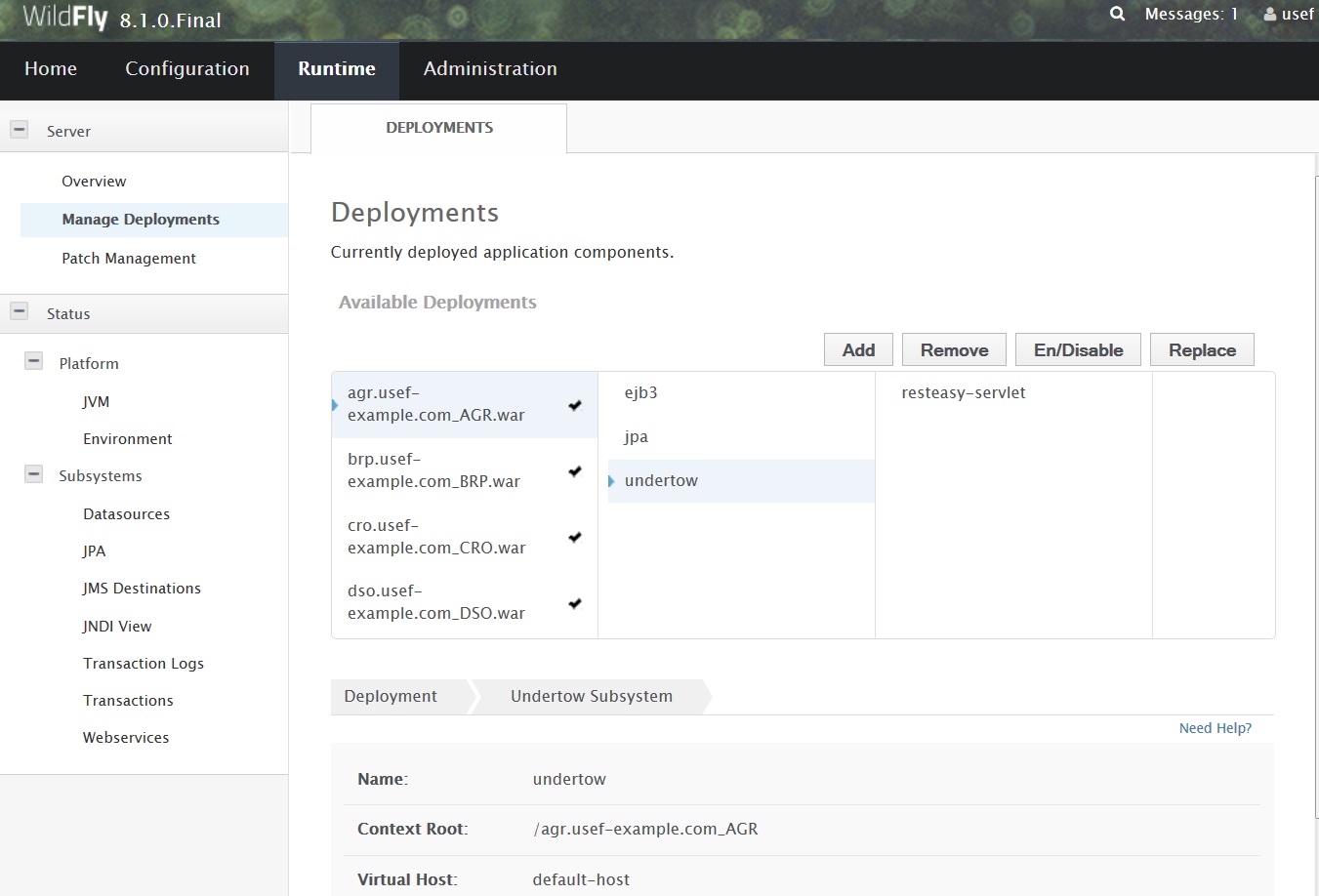
<?xml version="1.0" encoding="UTF-8"?>  
<TestMessage><MessageMetadata   
 SenderDomain="dso1.usef-example.com"   
 SenderRole="DSO"   
 RecipientDomain="cro1.usef-example.com"   
 RecipientRole="CRO"   
 TimeStamp="2014-10-14T09:30:47-05:00"  
 MessageID="00000000-0010-0000-0000-000000000001"   
 ConversationID="00000000-0000-0000-0000-000000000000"   
 Precedence="Routine"  
 ValidUntil="2014-11-14T09:30:47-05:00"/>  
</TestMessage>

For example, use Postman to send the message to CRO1 from AGR1.

You can check the console or logging to see if the message is sent.

## JBoss Management Console

The JBoss Management Console can be found on URL <http://localhost:9990>  
with user **usef** and password **usef**

Click on the tab “Runtime” and then “Manage Deployments” to check if the war files are deployed:  


At the bottom the value of “Context Root” is displayed, which is in this case “/agr1.usef-example.com\_AGR”.

Messages can be sent to the URL:

<https://agr1.usef-example.com:8443/agr1.usef-example.com_AGR/rest/MessageService/sendMessage>

## Nodes folder

For every node which has been configured in the usef-environment.yaml file, the usef-environment/nodes folder contains a folder with the running configuration and files which are used by USEF and JBoss Wildfly. By default, the usef-environment/nodes folder contains the following folders:

* + - configuration: The configuration folder with various configuration files.
    - content: Folder which contains static files.
    - data: The H2 database file is located in this folder.
    - deployments: The WAR file per participant is deployed via this folder.
    - lib: Library folder.
    - log: Folder which contains per participant the log-files.
    - temp: Temporary folder.

Do not change the content of these folders if you are not an advanced user!

# Configuration

## Adjusting the usef-environment.yaml configuration

By default, the usef-environment.yaml file (located in the usef-environment/config folder) contains the configuration for 8 USEF participants (3 AGRs, 2 BRPs, 1 CRO, 1 DSO and 1 MDC) for node ‘localhost’. The usef-environment.yaml file can be opened with a text editor, like Notepad++. At the bottom of the file, the domains and its participants are defined.

The usef-environment.yaml file describes all configuration items, which are explained in Appendix 7 and in the file itself.  
The standard value of ‘recipient\_endpoint’ assumes no proxy is used. If you are configuring a proxy, this entry should be removed.

If you change the usef-environment.yaml or the standalone-usef.xml file, then:

* + - Stop the USEF environment by manually press Ctrl+C in the console.
    - Run the cleanup script.
    - Run the prepare script
    - Run the start-h2-database script
    - Run the start-usef-environment script

Check if the configuration affects the behavior of USEF (Notice that the log files and folder structure can be checked to see if the changed parameters had effect, e.g. are all roles deployed).

## Logging

The USEF Reference Implementation uses the LogBack framework for application logging. Information exchanged may contain confidential information that must not be logged to the default log files. The LogBack configuration facilitates this.

For more information on this framework, please visit (<http://logback.qos.ch/>).

The logging configuration of USEF Reference Implementation contains information on e.g. log file locations and log levels. Each deployment type has its own logging configuration files, located in the deployment folder.

Below is an example of where these files are located:

* + - usef-deployments/usef-deployment-agr/src/main/resources/logback.xml and
    - usef-deployments/usef-deployment-agr/src/main/resources/LogBackAgr.xml

Note that the filters specified should not be modified, because that could result in a non-compliant USEF implementation.

After installation of the USEF Reference Implementation, each deployment has its own logging configuration XML file that enables you to change the log levels for this specific deployment.

Below is an example of where this file is located:

* + - usef-environment/nodes/localhost/configuration/agr1.usef-example.com\_AGR/LogBackWithAdditionalLoggers.xml

Application Server logging for generic non-application logging is configured in

* + - usef-environment/template standalone-usef.xml

## Setting up the database with Common Reference information

To create a consistent common reference between all participants, the participants need to be setup properly. The database of each BRP, AGR and DSO participant must be populated with consistent data in the correct tables before the connection and congestion point information is sent to the CRO. The “Publish Connections” process of each participant sends this information to the Common Reference Operator. Each participant will then query the Common Reference Operator to retrieve relevant common reference information.

This process and the related database models of all participants are described in [1].

The USEF Reference Implementation includes an example sql-script that can be used to populate the databases of all participants defined in the usef-environment.yaml file.

This sql script is located in:

* + - usef-environment/config/usef\_common\_reference.sql

*The example scenario is based on the existence of one MDC, one DSO, two BRPs and three AGRs. The DSO has defined four congestion points. BRP1 is engaged with AGR1 and AGR2, and BRP2 is engaged with AGR3.*

*The connections are all assumed to be home prosumers, with the allocation of homes to AGRs and congestion points defined below. There are 170 homes without AGR and 100 homes with AGR1 or AGR2 that are not on a congestion point. All homes allocated to congestion point 2 have PV which will be used to simulate deviations during the operate phase. For all connections associated with congestion point 4 there is no aggregator, which will lead to orange regime processing in case of congestion.*

*The example common reference contains:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *DSO1* | *CP1 (40)* | *CP2 (100 PV)* | *CP3 (400)* | *CP4 (20)* | *No CP (100)* |
| *BRP1 & AGR1* | *0* | *10* | *100* |  | *50* |
| *BRP1 & AGR2* | *10* | *40* | *100* |  | *50* |
| *BRP2 & AGR3* | *10* | *20* | *100* |  |  |
| *No AGR* | *20* | *30* | *100* | *20* |  |

*Note: an additional script with a smaller amount of connections (66 instead of 660) has been provided in:*

* + - *usef-environment/config/usef\_common\_reference\_small.sql*

## Reserved TCP ports

JBoss Wildfly and the H2 database are using TCP ports which are fixed. So, if any application or process is running on these TCP ports, USEF will fail to start. When the USEF environment is started, the TCP ports will be checked and report the PID of the process which is using the TCP port, if so.

Please, make sure the following TCP ports are reserved for USEF:

* 443: the HTTPS port, used by the Apache proxy (optional)
* 8082: the HTTP port, used by the H2 database console
* 8443: the HTTPS port, used by JBoss Wildfly
* 9092: the H2 database
* 9990: the JBoss management console HTTP port
* 9993: the JBoss management console HTTPS port

## Configure resolver entries

For participants to be able to reach each other, their domain names must resolve to a valid network address. For example, if the domains, “agr1.usef-example.com” and “dso1.usef-example.com” are hosted by the local machine, then those two domains must resolve to the IP address of the local machine. You can also specify a separate domain where Wildfly will be running.

If you installed BIND, the IP addresses will be resolved for you, otherwise – for demo purposes – you can work around this by adding the following lines to “/etc/hosts” (or “C:\Windows\System32\drivers\etc\hosts” on Windows):

127.0.0.1 agr1.usef-example.com  
127.0.0.1 agr2.usef-example.com  
127.0.0.1 agr3.usef-example.com  
127.0.0.1 brp1.usef-example.com  
127.0.0.1 brp2.usef-example.com  
127.0.0.1 cro1.usef-example.com  
127.0.0.1 dso1.usef-example.com  
127.0.0.1 mdc1.usef-example.com  
127.0.0.1 jboss.usef-example.com

## Configuring a proxy server (optional)

The Apache configuration must be modified in the following ways. Instead of “usef-example.com” use your own registered domain. As an example, the following configuration steps can be followed:

* + - Create the file /etc/apache2/conf-available/usef-proxy.conf with the following contents:

<VirtualHost \_default\_:443>

# This VirtualHost's identity

ServerAdmin webmaster@usef-example.com

ServerName www.usef-example.com

ServerAlias \*.usef-example.com

# Enable SSL proxy

SSLEngine on

SSLProxyEngine on

# SSLProxyCheckPeerName can be set to off for test purposes

SSLProxyCheckPeerName on

# SSL certificate files

SSLCertificateKeyFile "/etc/apache2/usef/usef.key"

SSLCertificateFile "/etc/apache2/usef/usef.crt"

# Proxy matching rules

# matching URLs for testing purposes

ProxyPassMatch ^/(.\*\.usef-example\.com\_.\*)$ https://jboss.usef-example.com:8443/$1

# matching USEF compliant URLs

ProxyPassMatch ^/USEF/2014/I/(.\*)/(.\*)/SignedMessage$ \

https://jboss.usef-example.com:8443/$1\_$2/rest/ReceiverService/receiveMessage

</VirtualHost>

* + - Create the folder /etc/apache2/usef as follows:

sudo mkdir /etc/apache2/usef

* + - Generate or obtain a private key and certificate and place them as usef.key and usef.crt in /etc/apache2/usef.
    - Enable all configuration items and restart apache:

sudo a2enmod ssl proxy proxy\_http

sudo a2ensite usef-proxy

sudo service apache2 restart

Please note that in the environment.yaml file, the recipient\_endpoint parameter must be removed to enable use of the proxy. This is documented in the section ‘Configuration’. This will also enable full USEF compliant URLs.

# Resolving participant information

Every participant participating in USEF must be able to resolve information of every other participant it communicates with. This information consists of the IP address and a public key for sealing and unsealing messages.

## Secure information provision using DNSSEC

To guarantee using DNSSEC for secure communication between participants, a recursive DNS server must be configured which is dedicated to USEF DNS queries. To be USEF compliant, the path to this server must be 100% trusted and the server must only allow secure DNSSEC responses from a participant’s DNS server.

With Bind version 9.9.5, this can be configured by defining option ‘dnssec-must-be-secure <domain> yes;’ in the options section which will only accept secure DNSSEC responses from the hierarchy starting with <domain>.

## By default, the configuration of Bind as described earlier in ‘ISC BIND

Installing ISC BIND 9.10.x is optional in demo environment, mandatory in production’ will have this option enabled already.

## Configuring DNS in wildfly

By default, Wildfly is configured to use a name server which is located on localhost (127.0.0.1). When the file participants\_dns\_info.yaml is used to resolve participant keys and IP addresses of participants are registered in the hosts file, this will still work without local DNS server.

If a DNS server is to be used, the willdfly.properties file has to be changed. This file can be found in folder usef-environment/template. The property dns.server=127.0.0.1 can then be changed to refer to the IP address of the actual trusted DNS server. Changes to this file will be taken into account when the USEF environment is rebuilt by using the prepare script.

For detailed information on USEF DNS configuration see Chapter 6.2 of [3].

## Resolving without DNS server

For demonstration and testing purposes, the file participants\_dns\_info.yaml is automatically configured for the participants which are deployed by default. This file contains the public key for sealing and unsealing and can be found in the generated folder usef-environment/nodes/localhost/configuration after the prepare script is executed.

In addition to the public key for sealing and unsealing and in absence of a DNS server, the local resolver library must be able to resolve the names of the participants to valid IP addresses. An example of how to do this be defining entries in the system’s hosts file is described in ‘

Configure resolver entries’.

# Appendix – Configuration of scheduled processes

The usef-environment.yaml file (located in the usef-environment/config folder) contains the configuration of all USEF roles and processes. This includes scheduled times for processes that are time triggered. This appendix explains the default configuration for these scheduled times and how they are related to each other.

For ease of understanding, these scheduled times are explained in two parts:

* + - Extraday triggers: triggers that are fired once a day or even less frequently. These triggers are typical for the processes in plan, validate (excluding flex trading) and settlement phases.
    - Intraday triggers: triggers that are fired multiple times per day. These triggers are typical for the flex trading processes and processes in the operate phase.

## Extraday triggers

The table below describes all processes that are triggered once a day or less frequently, to which roles they apply, which configuration parameters are involved and the default values. Also a remark is included to indicate if there is a relation to other processes and/or configuration parameters.

Each value is defined as the time on which the trigger is fired. If a configuration parameter is defined for multiple roles, the “{role}” is replaced by the applicable role in the configuration file.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Process** | **AGR** | **BRP** | **DSO** | **MDC** | **yaml configuration parameters** | **remark** |
| Initialize Non-UDI Clusters | 00:01 |  |  |  | agr\_initialize\_non\_udi\_time\_of\_day |  |
| Common Reference Query |  |  |  | 05:00 | mdc\_common\_reference\_query\_time |  |
| Common Reference Update | 11:00 | 11:00 | 11:00 |  | {role}\_common\_reference\_update\_time |  |
| Initialize Planboard | 12:00 | 12:00 | 12:00 |  | {role}\_initialize\_planboard\_time {role}\_initialize\_planboard\_days\_interval brp\_initialize\_planboard\_days\_ahead | The common reference update of all AGRs, BRPs and DSOs must be finished before the planboards are initialized to get consistent results. |
| Collect Forecast (DSO) |  |  | 13:00 |  | dso\_connection\_forecast\_time dso\_connection\_forecast\_days\_interval | The DSO needs to generate its non-aggregator forecast before it is able to process D-prognoses. Therefore the DSO Collect Forecast process must be scheduled earlier than the AGR equivalent. |
| Collect Forecast (AGR) | 13:30 |  |  |  | agr\_connection\_forecast\_time agr\_connection\_forecast\_days\_interval |  |
| Finalize A-Plan (AGR) | 19:30 |  |  |  | day\_ahead\_gate\_closure\_time - agr\_finalize\_aplans\_ptus\_before\_gate\_closure \* ptu\_duration | The AGR must finalize A-plans and/or start the validate phase before the DSO can create missing D-prognoses. |
| Create Missing Prognoses |  | 20:00 | 20:00 |  | day\_ahead\_gate\_closure\_time - day\_ahead\_gate\_closure\_ptus \* ptu\_duration |  |
| Finalize A-Plan (BRP) |  | 21:00 |  |  | day\_ahead\_gate\_closure\_time - brp\_finalize\_aplans\_ptus\_before\_gate\_closure \* ptu\_duration | The BRP can only finalize A-plans after it has created any missing A-plans. |
| Day Ahead Gate Closure Time | 22:00 | 22:00 | 22:00 |  | day\_ahead\_gate\_closure\_time | All participants must agree on this time. |
| Initiate Settlement Time | 01:00 | 10:00 | 11:00 |  | {role}\_initiate\_settlement\_time {role}\_initiate\_settlement\_day\_of\_month |  |
| Dispute Settlement Without Response |  | 22:00 | 22:00 |  | {role}\_settlement\_message\_disposal\_time |  |
| Collect Orange Regime Data |  |  | 12:00 |  | dso\_initiate\_collect\_orange\_regime\_data\_time dso\_initiate\_collect\_orange\_regime\_data\_day\_of\_month |  |

## Intraday triggers

The table below describes all processes that are triggered multiple times a day, to which roles they apply, which configuration parameters are involved and the default values. Each value is defined as an initial delay, followed by the time interval between subsequent triggers (initial delay -> interval).

If a configuration parameter is defined for multiple roles, the “{role}” is replaced by the applicable role in the configuration file.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process** | **AGR** | **BRP** | **DSO** | **unit** | **yaml configuration parameters** |
| Flex Offer Schedule | 5 -> 900 |  |  | seconds | agr\_flexoffer\_initial\_delay\_in\_seconds -> agr\_flexoffer\_interval\_in\_seconds |
| Determine Net Demands | 5 -> 300 |  |  | seconds | agr\_determine\_netdemands\_initial\_delay\_in\_seconds -> agr\_determine\_netdemands\_interval\_in\_seconds |
| Control ADS | 5 -> 900 |  |  | seconds | agr\_control\_ads\_initial\_delay\_in\_seconds -> agr\_control\_ads\_interval\_in\_seconds |
| Flex Order |  | 5 -> 30 | 5 -> 900 | seconds | {role}\_flexorder\_initial\_delay\_in\_seconds -> {role}\_flexorder\_interval\_in\_seconds |
| Meter Data Query Expiration |  |  | 0 -> 60 | minutes | 0 -> dso\_meter\_data\_query\_expiration\_check\_interval\_in\_minutes |
| Operate |  |  | 1 -> 60 | seconds | dso\_operate\_initial\_delay\_in\_seconds -> dso\_operate\_interval\_in\_seconds |
| Identify Changes in Forecast | 5 -> 900 |  |  | seconds | agr\_identify\_change\_in\_forecast\_initial\_delay\_in\_seconds -> agr\_identify\_change\_in\_forecast\_interval\_in\_seconds=900 |
| Non-UDI Retrieve ADS Goal Realization | 15 -> 15 |  |  | minutes | agr\_non\_udi\_retrieve\_ads\_goal\_realization\_interval\_in\_minutes -> agr\_non\_udi\_retrieve\_ads\_goal\_realization\_interval\_in\_minutes |

1. http://www.bind9.net/arm910.pdf [↑](#footnote-ref-1)