Table of content

1 About this manual 2

1.1 About USEF 2

1.2 About the Reference Implementation 2

1.3 Document scope 2

1.4 Document conventions 3

1.5 How to use this document 3

1.6 More information 3

1.7 Terms and abbreviations 3

1.8 Licensing 4

2 Source code 5

2.1 Prerequisites 5

2.2 Naming conventions 5

2.3 Obtaining the source code 6

2.4 Libraries in use 6

3 Project structure 7

3.1 Service layer 7

3.2 Workflow layer 7

3.3 Pluggable Business Component layer 8

4 Application design 9

4.1 Service layer 9

4.2 Workflow layer 11

4.3 Pluggable Business Component layer 12

5 Tools 40

5.1 Database initializer 40

5.2 Message injector 40

6 Appendix - PBC implementation catalogue 41

6.1 PBC Feeder 41

6.2 Aggregator 43

6.3 Balance Responsible Party 52

6.4 Distribution System Operator 56

6.5 Meter Data Company 63

7 Appendix - PBC implementation manual 64

# About this manual

## 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, 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 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.

A layered flexibility is offered to allow USEF implementer candidates to pick that layer of the delivered code fit for purpose. Three main layers are therefore introduced in the reference implementation:

* Pluggable Business Component layer: this layer consists of implementations of business services that are not part of USEF itself, but are needed for a fully functional implementation of USEF. The Reference Implementation provides simple stub implementations of these business services that should be replaced by implementations of the USEF implementer’s business logic, or interfaces to existing systems providing that logic.
* Workflow layer: this layer provides an implementation of the USEF processes and the business services specified by USEF.
* Service layer: this layer provides a USEF compliant messaging framework, USEF message validation, encrypted databases and data models, logging functionality, code common to all roles and a set of business and workflow APIs.

By combining the workflow and the service layer, a USEF compliant implementation of the different roles is offered.

By adding the Pluggable Business Component layer, business components can be plugged in to setup and create a USEF compliant executable application for the different roles.

## Document scope

This document serves as a guide to start working with the reference implementation code. It wants to give an insight on the project structure per layer (Service, Workflow, and Pluggable Business Component), the application interfaces and application design of the major building blocks.

It contains an overview and description of all Pluggable Business Components that are required per role to set up an executable application as well as a description on how to write and use custom implementations of the Pluggable Business Components.

Although this document serves as a guide to start working with the code, it is not the goal to describe on code level how to develop a USEF implementation. As such, it is not the intention of this document to present a set of Java implementation guidelines or go into micro design details of the reference implementation.

## Document conventions

Code snippets and examples are shown in this format.

* <Role> indicates that this can be replaced by agr, dso, cro, brp or mdc.
* <Logical Name> indicates that in the code this is replaced by a business wise readable name.

## How to use this document

This document is structured in the following topics:

* **Source code**: this chapter gives information on how to get the code, what libraries are used and what naming conventions are applied.
* **Project structure**: this chapter elaborates on the project structure within each of the three layers and the dependencies between the projects.
* **Application design**: this chapter gives information on the interfaces and application design for the major building blocks in the three layers. In particular it contains a list and details of all Pluggable Business Components.
* **Tools**: the reference implementation is accompanied by a set of tools that are useful to prepare a USEF simulation run. This chapter lists these tools and where to find them.
* **PBC implementation catalogue**: this appendix lists all implementations of Pluggable Business Components that are delivered together with the reference implementation.
* **PBC implementation manual**: this appendix describes the steps to create a Pluggable Business Component implementation and override the default PBC implementation of the reference implementation.

## More information

Further information can be found in following documents:

* USEF specification 2015 on which the reference implementation is based.
* How to install USEF reference implementation: a step-by-step manual on how to install the USEF reference implementation for a simulation run.
* USEF reference System Architecture: a description of the architectural structure of the software systems that are built for the USEF Reference Implementation. This document describes the logical components of the solution, contains sequence diagrams to illustrate the interaction between these components and makes the link to the physical components realizing the logical components.
* Java doc: the generated Java documentation describing the interfaces and classes which can be found in the folder usef-javadoc of the delivery USEF reference implementation.

References:

|  |  |
| --- | --- |
| **Reference** | **Description** |
| [1] | USEF Specifications 2015 |
| [2] | USEF The Framework Implemented – Installation Manual |
| [3] | USEF The Framework Implemented – Implementation Guidelines |
| [4] | USEF The Framework Explained |

## Terms and abbreviations

Definitions of USEF specific terms and abbreviations can be found in [1] and [2].

## Licensing

The software source code of the reference implementation is provided by the USEF Foundation. The copyright and all other intellectual property rights relating to all software source code provided by the USEF Foundation (and changes and modifications as well as on new versions of this software source code) belong exclusively to the USEF Foundation and/or its suppliers or licensors. Total or partial transfer of such a right is not allowed. The user of the software source code made available by USEF Foundation acknowledges these rights and will refrain from any form of infringement of these rights.

The USEF Foundation provides this software source code “as is”. In no event shall the USEF Foundation and/or its suppliers or licensors have any liability for any incidental, special, indirect or consequential damages; loss of profits, revenue or data; business interruption or cost of cover or damages arising out of or in connection with the software source code or accompanying documentation.

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

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

# Source code

## Prerequisites

All code is written according JEE8 standards and should as such be ready to load in any mature Java IDE.

The code makes no use of proprietary application server libraries.

A relational database is used to store the operational data. To support multiple databases no database logic (e.g. stored procedures) is introduced in the application. To be USEF compliant, the chosen database must support database encryption. The reference implementation is delivered with a H2 database.

The repository contains a folder usef-environment which contains a lib folder. When using the USEF tooling, it determines on which OS you are running USEF and uses the OS specific library.

## Naming conventions

All package names start with energy.usef and consist of logical singulars without capitals.

Example: energy.usef.core.service.endpoint

The following class names reflect the business and/or technical purpose.

* Endpoint: <Logical Name>Endpoint
* Message driven bean: <Logical Name>MDB
* Business service: <Logical Name>BusinessService
* Trigger: <Logical Name>Trigger
* Helper: <Logical Name>Helper
* Utility: <Logical Name>Util
* Enumeration types: <Logical Name>Type
* Repository: <Logical Name>Repository
* Data Transfer Object: <Logical Name>DTO

Example: OutgoingQueueMDB, MessageFilterService

Database table names are defined as logical singulars in capitals and split with underscore symbols if a name is composed of different names.

Example: COMMON\_REFERENCE\_DOMAIN

Persistent bean class names correspond to the table they represent, following the normal writing conventions.

Example: CommonReferenceDomain

XML bean class names correspond to the XML root element.

Example: <?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"yes\"?><TestMessage>..... results in TestMessage

## Obtaining the source code

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

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

## Libraries in use

* JUnit, version 4.11
* Mockito, version 1.9.5
* Powermock, version 1.5.5
* Resteasy, version 3.0.6.Final
* dnsjava, version 2.1.6
* kalium, version 0.2.1
* Google-http-client, version 1.19
* Snakeyaml, version 1.15
* SLF4J, version 1.7.2
* Logback, version 1.1.2
* Reflections, version 0.9.9-RC1
* Apache commons-codec, version 1.9
* Apache commons-lang, version 3.3.2
* Apache commons-io, version 2.4
* DBUnit, version 2.5.0
* XMLUnit, version 1.5
* Joda time, version 1.6.2
* Janino, version 2.5.10
* Recurrent version 0.3.3

# Project structure

The list below shows the mapping between the project structure and the layers. In the following sub chapter more detail is presented for each of the three layers.

* The service layer is contained in usef-core. All package names within usef-core start with energy.usef.core. There are no dependencies towards projects in the other layers.
* The workflow layer is contained in usef-workflow. All package names within usef-workflow start with energy.usef. Following dependencies are made to the service layer:
* usef-core-transport
* usef-core-workflow
* usef-core-planboard
* The pluggable business component layer is contained in usef-simulation. All packages within usef-simulation start with energy.usef. There are no dependencies on this level towards projects in the other layers.

## Service layer

This layer contains projects that define artifacts that are common to all USEF roles. The following projects are included in this layer:

* usef-core-api: contains all common interfaces.
* usef-core-commons: contains all kind off common classes used like utility, base abstract and exception classes. A dependency is made with:
* usef-core-api
* usef-core-planboard: contains all common classes and utilities related to the planboard. A dependency is made with:
* usef-core-transport
* usef-core-transport: contains all classes that are related to message exchange between USEF participants. A dependency is made with:
* usef-core-commons
* kalium - org.abstractj.kalium
* dnsjava – dnsjava
* Google http client - com.google.http-client
* Google reflections - org.reflections
* usef-core -workflow: contains the base classes that are used in the role specific workflow projects. A dependency is made with:
* usef-core-commons
* Google reflections - org.reflections

## Workflow layer

This layer contains projects that implement the USEF processes and are Role specific.

* usef-agr: contains an implementation of the AGR workflows.
* usef-brp: contains an implementation of the BRP workflows.
* usef-cro: contains an implementation of the CRO workflows.
* usef-dso: contains an implementation of the DSO workflows.
* usef-mdc: contains an implementation of the MDC workflows.

None of them introduce new dependencies upon those already defined in usef-workflow.

## Pluggable Business Component layer

This layer will contain projects that offer implementations of the Pluggable Business Components required to simulate a USEF run.

* usef-simulation-agr: implementations of required Pluggable Business Components in the AGR workflows. Depends on:
* usef-agr
* usef-simulation-brp: implementations of required Pluggable Business Components in the BRP workflows. Depends on:
* usef-brp
* usef-simulation-dso: implementations of required Pluggable Business Components in the DSO workflows. Depends on:
* usef-dso
* usef-simulation-mdc: implementations of required Pluggable Business Components in the MDC workflows. Depends on:
* usef-mdc

# Application design

## Service layer

### Confidential logging

Logging is performed by using SLF4J (Simple Logging Facade for Java). By using the default mechanism, sensitive data is not written to the log files to comply with the USEF privacy and security guidelines. The reference implementation includes a dedicated confidential logger to write the confidential information to separate log files for e.g. debugging purposes. Statements to the confidential logging on other levels than debug will not be logged to the confidential logger.

See [3] for the configuration of the confidential logger.

To use the confidential logger add in the declaration section a private static Logger:

private static final Logger LOGGER\_CONFIDENTIAL = LoggerFacto-ry.getLogger(USEFLogCategory.CONFIDENTIAL);

And add in the code where this confidential logging is wanted:

LOGGER\_CONFIDENTIAL.debug("The text describing the log", the message to log);

### Exception handling

Exceptions are thrown up until the final class where they are handled properly.

usef-core-commons contains a definition of a business and technical exception that are to be used.

* energy.usef.core.exception.BusinessException
* energy.usef.core.exception.TechnicalException

More specific exception classes are inherited from one of these two classes (depending if it concerns a business or technical exception extension).

### Database entity classes and schema

All database entity classes are placed in namespaces ending in ‘model’.

* usef-core-transport project, energy.usef.core.model package: common message transport database entities.
* usef-core-planboard project, energy.usef.core.model package: common planboard database entities.
* usef-agr project, energy.usef.agr.model package: AGR specific database entities.
* usef-cro project, energy.usef.cro.model package: CRO specific database entities.
* usef-dso project, energy.usef.dso.model package: DSO specific database entities.
* usef-brp project, energy.usef.brp.model package: BRP specific database entities.
* usef-mdc project, energy.usef.mdc.model package: MDC specific database entities.

Each role has its own database schema. This schema is generated at deployment time based on a persistence.xml file identifying the database entity classes to use per role. This file can be found per role is in the usef-deployment-<Role> projects in the src/main/resources/META\_INF folder.

### Message interface and channels

As described as a recommended practice in the messaging chapter of the USEF specification, all messages between 2 participants are exchanged via the same REST channel with the same interface.

* The interface is defined in energy.usef.core.data.xml.bean.message.Message.
* Messages are sent via the energy.usef.core.service.rest.sender.SenderService.
* Messages are received via the energy.usef.core.endpoint.ReceiverEndpoint.

### USEF message classes

The messages exchanged between the different actors are described in the USEF specifications. The message format is defined in an XSD file. The XSD is included in the usef-core-api project in the src/main/resources/xsd folder.

Java message classes are generated based on this XSD at compile time and are placed in the energy.usef.core.data.xml.bean.message package in the target/generated-sources/xjc folder of the usef-core-api project.

### Message security and validations

As described in the USEF specifications and in the USEF reference System Architecture, different steps have to be performed before sending and accepting a message. These steps are coordinated by the classes SenderService and ReceiverEndpoint mentioned above.

Classes of interest that to support these steps are (see the java doc for more information):

* energy.usef.core.util.encryption.NaCl
* energy.usef.core.service.business.IncomingMessageVerificationService
* energy.usef.core.service.business.MessageEncryptionService
* energy.usef.core.service.business.ParticipantDiscoveryService

### Inbound message controller framework

Per participant, all valid inbound messages are put on the same persistent queue. The messages of this queue must be delivered to the corresponding business service(s).

For this, an inbound message controller framework is designed to distribute incoming xml messages to the correct business service. The steps performed by the framework are as follows:

* A valid inbound message is put on the inbound persistent queue.
* The inbound queue is read by the energy.usef.core.service.mdb.IncomingQueueMDB bean.
* The message is passed to the energy.usef.core.service.helper.DispatcherHelperService that will route the message to the corresponding business controller based on the message type. All controllers extend the energy.usef.core.controller.BaseIncomingMessageController class.
* The controller will on its turn invoke the required business services.

The business controllers are put in the corresponding usef-<Role> project within the energy.usef.<Role>.controller package.

Example: usef-cro project; energy.usef.cro.controller.CommonReferenceQueryController

### Outbound message controller framework

Exponential backoff is an algorithm that retries requests to the server based on certain status codes in the server response. The retries exponentially increase the waiting time up to a certain threshold. The idea is that if the server is down temporarily, it is not overwhelmed with requests hitting at the same time when it comes back up. See [3] for the configuration and details of this mechanism.

Outbound messages that could not be delivered or for which no response was received in a timely manner, sometimes need special treatment to e.g. bring the failure to the attention of an operator. For this a similar framework is implemented as the inbound message controller framework. The steps performed by the framework are as follows:

* Not delivered messages or messages without a timely received response that need further treatment are put on a dedicated persistent queue.
* The queue is read by the energy.usef.core.service.mdb.NotSentQueueMDB.
* The message is passed to the energy.usef.core.service.helper. OutgoingErrorMessageDispatcherHelperService that will route the message to the corresponding OutgoingErrorMessageController based on the message type. All controllers extend the energy.usef.core.controller.error.BaseOutgoingErrorMessageController class.
* The controller will on its turn execute the required business logic.

The outgoing error controllers are put in the corresponding usef-<Role> project within the energy.usef.<Role>.controller.error package.

Example: usef-dso project; energy.usef.dso.controller.error.FlexOrderErrorController

## Workflow layer

### Base classes

The USEF reference System Architecture shows a sequence diagram of a common approach to implement the USEF workflows. To realize this, base workflow classes are defined in usef-core-workflow that are extended when a workflow step is realized. These base classes are (see the Javadoc for more information):

* energy.usef.core.workflow.Event
* energy.usef.core.workflow.BaseEventController
* energy.usef.core.workflow.BaseWorkflowCoordinator
* energy.usef.core.workflow.CoordinatorWorkflowContext

An implementation of a Pluggable Business Component is loaded via reflection based on configuration. The class that realizes this is energy.usef.core.workflow.step. WorkflowStepLoader.

### Time based triggers

For time based triggers, JEE7 ScheduledExecutorService is used. For this, each trigger class must register itself during startup in order to be scheduled. This subscription mechanism is delivered by the helper class energy.usef.core.service.helper.SchedulerHelperService.

### Workflow realizations

As mentioned in the project structure chapter, each USEF role has its dedicated project that will contain all the code for the role specific workflow realizations.

The USEF reference System Architecture shows a mapping of each process coordinator on the USEF business flows as defined in the USEF specifications. All related classes for one process coordinator are placed in a package with a name corresponding to the USEF step in the process flows.

Example: energy.usef.dso.workflow.validate.gridsafetyanalysis

## Pluggable Business Component layer

### Pluggable Business Component interface

The USEF reference System Architecture shows a sequence diagram of a common approach to invoke a Pluggable Business Component in a USEF workflow. This document also lists all the assumptions made on implementations of Pluggable Business Component.

The interface to implement and to invoke a Pluggable Business Component is defined in energy.usef.core.workflow.WorkflowStep (see the java doc for more information).

The next chapter lists all Pluggable Business Components and the in- and output parameters exchanged via the WorkflowContext object.

PBCs use two types of input and output parameters that can be placed in the WorkflowContext:

* Simple Java objects (e.g. List, int, Long, boolean)
* USEF specific Data Transfer Objects (DTOs), tailored for a specific situation where the other two types are not applicable. Consult the java doc for the object description, which can be found in folder usef-javadoc of the delivery USEF reference implementation.

### Pluggable Business Components

This chapter contains a list of Pluggable Business Component (PBC) specifications required to run USEF processes. Each Pluggable Business Component is positioned within the USEF process flows.

The appendix ‘PBC stub implementation catalogue’ lists the simple stub implementations provided with the Reference Implementation of these business services.

The USEF reference System Architecture contains a description of the different process coordinators and identifies what Pluggable Business Components are required per process coordinator.

#### Aggregator

##### Create Elements

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Plan | NA | Retrieve Congestion Points and BRPs | |
| **Mapping name** | AGR\_UPDATE\_ELEMENT\_DATA\_STORE |
| **Context** | An Aggregator initializes its plan for a number of days in advance and stores connection forecast information. First step is to update the element data store in order to supply up-to-date data to the subsequent portfolio initialization process.  This Pluggable Business Component fills the element data store with the details of the portfolio elements associated with the aggregator’s connections.  The PBC receives as input the day for which the profile has to be created, the duration of a PTU expressed in minutes, and a list of connections to create the profile for. The output is a list of elements containing profile power values. |
| **Input** | * org.joda.time.LocalDate PERIOD   The day for which elements will be created.   * int PTU\_DURATION   The duration of a PTU expressed in minutes.   * List <energy.usef.agr.dto.ConnectionPortfolioDto > CONNECTION\_PORTFOLIO\_DTO\_LIST   List of connections to create the elements for. |
| **Output** | * List <energy.usef.agr.dto.ElementDto > ELEMENT\_LIST   The created element list |
| **References** | See [2]: chapter 5.2.3 AgrUpdateElementDataStoreCoordinator |

##### Create Profile

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Plan | NA | Retrieve Congestion Points and BRPs | |
| **Mapping name** | AGR\_CREATE\_CONNECTION\_PROFILE |
| **Context** | An Aggregator initializes its plan for a number of days in advance and stores connection forecast information.  This Pluggable Business Component reads the load profiles for each PTU from the element data store and populates the AGR portfolio with profile power values.  The PBC receives as input the day for which the profile has to be created, the duration of a PTU expressed in minutes, a list of connections to create the profile for and a map containing the elements that belong to these connections. The output is a list of connections containing profile power values. |
| **Input** | * org.joda.time.LocalDate PERIOD   The day for which the forecast has to be created.   * int PTU\_DURATION   The duration of a PTU expressed in minutes.   * List <energy.usef.agr.dto.ConnectionPortfolioDto > CONNECTION\_PORTFOLIO\_DTO\_LIST   List of connections to create the forecast for.   * Map<String, List<energy.usef.agr.dtoElementDto>> ELEMENT\_PER\_CONNECTION\_MAP   Map containing the elements for the specified connections. |
| **Output** | * List <energy.usef.agr.dto.ConnectionPortfolioDto > CONNECTION\_PORTFOLIO\_DTO\_LIST   List of connections from input, expanded with profile power values. |
| **References** | See [2]: chapter 5.2.4 AgrCreateConnectionProfileCoordinator |

##### Create UDI

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Plan | NA | Retrieve Congestion Points and BRPs | |
| **Mapping name** | AGR\_CREATE\_UDI |
| **Context** | An Aggregator initializes its plan for a number of days in advance and stores connection forecast information.  This Pluggable Business Component reads the element data store and creates UDIs, UDI events and capabilities for them in the portfolio.  The PBC receives as input the day for which the UDIs have to be created, the duration of a PTU expressed in minutes, a list of connections to create the UDIs for and a map containing the elements that belong to these connections and on which the UDIs shall be based. The output is a list of connections containing UDIs and a map with all UDI endpoints and associated UDI events and capabilities that can be used to control the UDI. |
| **Input** | * org.joda.time.LocalDate PERIOD   The day for which the forecast has to be created.   * int PTU\_DURATION   The duration of a PTU expressed in minutes.   * List <energy.usef.agr.dto.ConnectionPortfolioDto > CONNECTION\_PORTFOLIO\_DTO\_LIST   List of connections to create UDIs for.   * Map<String, List< energy.usef.agr.dto.ElementDto>> ELEMENT\_PER\_CONNECTION\_MAP   Map containing the elements for the specified connections. |
| **Output** | * List <energy.usef.agr.dto.ConnectionPortfolioDto > CONNECTION\_PORTFOLIO\_DTO\_LIST   List of connections from input, expanded with UDIs   * Map<String, List< energy.usef.agr.dto.UdiEventDto>> UDI\_EVENTS\_PER\_UDI\_MAP   Map containing the UDI endpoints and the associated UDI events and capabilities that can be used to control the UDI. |
| **References** | See [2]: chapter 5.2.5 AgrCreateUdiCoordinator |

##### Collect Forecast

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Plan | NA | Collect Forecast | |
| **Mapping name** | AGR\_CREATE\_N\_DAY\_AHEAD\_FORECAST |
| **Context** | An Aggregator initializes its plan for a number of days in advance and stores connection forecast information.  This Pluggable Business Component creates a load forecast for each PTU for all specified connections by adding forecast power values to the AGR portfolio for all connections. Additionally, the forecast per DTU for all UDIs is added to the portfolio.  The PBC receives as input the day for which the forecast has to be created, the duration of a PTU expressed in minutes and a list of connections to create the forecast for. The output is a list of connections containing forecast power values on connection and UDI level. |
| **Input** | * org.joda.time.LocalDate PTU\_DATE   The day for which the forecast has to be created.   * int PTU\_DURATION   The duration of a PTU expressed in minutes.   * List <energy.usef.agr.dto.ConnectionPortfolioDto> CONNECTION\_PORTFOLIO   List of connections to create the forecast for. |
| **Output** | * List <energy.usef.agr.dto.ConnectionPortfolioDto > CONNECTION\_PORTFOLIO   List of connections from input, expanded with collected forecast power values on connection and UDI level. |
| **References** | See [2]: chapter 5.2.6 AgrConnectionForecastPlanBoardCoordinator and 5.2.7 AgrUpdateConnectionForecastCoordinator.  See [1]: The need and the usage of Aggregator forecasts is elaborated in sections 2.3.1.1. |

##### Non-UDI Collect Forecast

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Plan | NA | Collect Forecast | |
| **Mapping name** | AGR\_CREATE\_NON\_UDI\_N\_DAY\_AHEAD\_FORECAST |
| **Context** | An Aggregator initializes its plan for a number of days in advance and stores connection forecast information.  This Pluggable Business Component creates a load forecast for each PTU for all specified connections by adding forecast power values to the AGR portfolio for all connections.  The PBC receives as input the day for which the forecast has to be created, the duration of a PTU expressed in minutes and a list of connections to create the forecast for. The output is a list of connections containing forecast power values on connection level. |
| **Input** | * org.joda.time.LocalDate PTU\_DATE   The day for which the forecast has to be created.   * int PTU\_DURATION   The duration of a PTU expressed in minutes.   * List <energy.usef.agr.dto.ConnectionPortfolioDto> CONNECTION\_PORTFOLIO   List of connections to create the forecast for. |
| **Output** | * List <energy.usef.agr.dto.ConnectionPortfolioDto > CONNECTION\_PORTFOLIO   List of connections from input, expanded with collected forecast power values.. |
| **References** | See [2]: chapter 5.2.6 AgrConnectionForecastPlanBoardCoordinator and 5.2.7 AgrUpdateConnectionForecastCoordinator.  See [1]: The need and the usage of Aggregator forecasts is elaborated in sections 2.3.1.1. |

##### Optimize AGR Portfolio

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Plan, Operate | NA | Re-optimize portfolio | |
| **Mapping name** | AGR\_REOPTIMIZE\_PORTFOLIO |
| **Context** | For several reasons, an Udi aggregator may re-optimize its portfolio (for example, to maximize its income or minimize prosumer complaints) by influencing consumption or production of the devices (via ADS) behind the connections.  This Pluggable Business Component is specific for Udi aggregators and re-optimizes the complete portfolio for a period and returns an updated connection portfolio, as well as a list of device messages, specifying the actions to be taken to optimize the portfolio and stored in the portfolio database.  The PBC is triggered when the portfolio changes (flex orders, changes in forecast or deviations in the Operate phase). The input for this PBC is the duration of a PTU expressed in minutes, the current PTU index, the date, the complete connection portfolio for the specified date, as well as all planboard documents relevant to the optimization process:   * The most recent D-prognoses and A-plans * All unprocessed FlexOrder messages for the specified date * All Prognosis messages that the above FlexOrder messages are based on * All relevant UDI Events with capabilities   The output is re-optimized connection portfolio and a list of device messages to send via ADS to the devices behind the connection to optimize the portfolio based on the capabilities defined in the supplied UDI events. |
| **Input** | * int PTU\_DURATION   The duration of a PTU expressed in minutes.   * org.joda.time.LocalDate PTU\_DATE   The date for which the portfolio needs to be re-optimized.   * int CURRENT\_PTU\_INDEX   Current PTU index.   * List <energy.usef.agr.dto.ConnectionPortoflioDTO> CONNECTION\_PORTFOLIO\_IN   Connection portfolio to be re-optimized.   * Map <String, List <String>> CONNECTION\_GROUPS\_TO\_CONNECTIONS\_MAP   Map of connection group identifiers and associated connection identifiers.   * List <energy.usef.core.workflow.dto.Prognosis> LATEST\_A\_PLAN\_DTO\_LIST   List of most recent A-plans.   * List <energy.usef.core.workflow.dto.Prognosis> LATEST\_D\_PROGNOSIS\_DTO\_LIST   List of most recent D-prognoses.   * List <energy.usef.core.workflow.dto.FlexOrderDto> RECEIVED\_FLEXORDER\_LIST   List of all FlexOrders for the period.   * List <energy.usef.core.workflow.dto.PrognosisDTO> RELEVANT\_PROGNOSIS\_LIST   List of all prognosis messages that are relevant to the FlexOrder messages for the period   * List <energy.usef.agr.dto.UdiEvent> UDI\_EVENTS   List of all UDI Events associated with the connections in the portfolio. |
| **Output** | * List <energy.usef.agr.dto.udi.DeviceMessageDto> UDI\_CONTROL\_MESSAGE\_DTO\_LIST   A list of device messages containing device requests to send via ADS to the devices behind the connection to optimize the portfolio.   * List <energy.usef.agr.dto.ConnectionPortoflioDTO> CONNECTION\_PORTFOLIO\_OUT   Re-optimized connection portfolio. |
| **References** | See [2]: chapter 5.2.8 AgrReOptimizePortfolioCoordinator  See [1]: The process flow in the Plan phase is described in section 2.3.1.The process flow in the Operate phase is described in section 2.3.3. |

##### Non-UDI Optimize AGR Portfolio

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Plan, Operate | NA | Re-optimize portfolio | |
| **Mapping name** | AGR\_NON\_UDI\_REOPTIMIZE\_PORTFOLIO |
| **Context** | For several reasons, a non-Udi aggregator may re-optimize its portfolio (for example, to maximize its income or minimize prosumer complaints) by influencing consumption or production of the devices (via ADS) behind the connections.  This Pluggable Business Component is specific for non-Udi aggregators and re-optimizes the complete portfolio for a period and returns an updated connection portfolio.  The PBC is triggered when the portfolio changes (flex orders, changes in forecast or deviations in the Operate phase). The input for this PBC is the duration of a PTU expressed in minutes, the current PTU index, the date, the complete connection portfolio for the specified date, as well as all planboard documents relevant to the optimization process:   * The most recent D-prognoses and A-plans * All unprocessed FlexOrder messages for the specified date * All Prognosis messages that the above FlexOrder messages are based on   The output is re-optimized connection portfolio. |
| **Input** | * int PTU\_DURATION   The duration of a PTU expressed in minutes.   * org.joda.time.LocalDate PTU\_DATE   The date for which the portfolio needs to be re-optimized.   * int CURRENT\_PTU\_INDEX   Current PTU index.   * List <energy.usef.agr.dto.ConnectionPortoflioDTO> CONNECTION\_PORTFOLIO\_IN   Connection portfolio item to be re-optimized.   * Map <String, List <String>> CONNECTION\_GROUPS\_TO\_CONNECTIONS\_MAP   Map of connection group identifiers and associated connection identifiers.   * List <energy.usef.core.workflow.dto.Prognosis> LATEST\_A\_PLAN\_DTO\_LIST   List of most recent A-plans.   * List <energy.usef.core.workflow.dto.Prognosis> LATEST\_D\_PROGNOSIS\_DTO\_LIST   List of most recent D-prognoses.   * List <energy.usef.core.workflow.dto.FlexOrderDto> RECEIVED\_FLEXORDER\_LIST   List of all FlexOrders for the period.   * List <energy.usef.core.workflow.dto.PrognosisDTO> RELEVANT\_PROGNOSIS\_LIST   List of all prognosis messages that are relevant to the FlexOrder messages for the period |
| **Output** | * List <energy.usef.agr.dto.ConnectionPortoflioDTO> CONNECTION\_PORTFOLIO\_OUT   Re-optimized connection portfolio. |
| **References** | See [2]: chapter 5.2.8 AgrReOptimizePortfolioCoordinator  See [1]: The process flow in the Plan phase is described in section 2.3.1.The process flow in the Operate phase is described in section 2.3.3. |

##### New Prognoses Required

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Plan, Operate | NA | New A-Plan Or D-Prognosis Required? | |
| **Mapping name** | AGR\_RECREATE\_PROGNOSES |
| **Context** | After portfolio optimization, an Aggregator needs to decide whether a new A-plan and/or D-prognosis is required.  This Pluggable Business Component makes the decision whether to actually re-create A-plans and/or D-prognoses based on the current content of those plans and the state of the portfolio. It returns a list of A-plan and D-prognoses sequences that need to be re-created.  The input for this PBC is a list of latest Prognosis of type 'D-Prognosis', a list of latest Prognosis of type 'A-Plan', and the current (and hence latest) portfolio. The output is a list of sequence numbers of the A-Plans that will be re-created, and a list of sequence numbers of the D-Prognoses that will be re-created. |
| **Input** | * int PTU\_DURATION   The duration of a PTU expressed in minutes.   * org.joda.time.LocalDate PERIOD   The day for which the prognoses are checked.   * List <energy.usef.core.workflow.dto.PrognosisDto> LATEST\_D\_PROGNOSES\_DTO\_LIST:   List of latest Prognosis of type 'D-Prognosis'.   * List <energy.usef.core.workflow.dto.PrognosisDto> LATEST\_A\_PLANS\_DTO\_LIST :   List of latest Prognosis of type 'A-Plan'.   * List <energy.usef.agr.dto.ConnectionPortfolioDTO> CURRENT\_PORTFOLIO:   The current portfolio.   * Map <String, List <String>> CONNECTION\_GROUPS\_TO\_CONNECTIONS\_MAP   Map of connection group identifiers and associated connection identifiers. |
| **Output** | * List <Long> REQUIRES\_NEW\_APLAN\_SEQUENCES\_LIST:   List of sequence numbers of the A-Plans that will be re-created.   * List <Long>REQUIRES\_NEW\_DPROGNOSIS\_SEQUENCES\_LIST:   List of sequence numbers of the D-Prognoses that will be re-created. |
| **References** | See [2]: chapter 5.2.9 AgrReCreatePrognosesCoordinator  See [1]: The process flow in the Plan phase is described in section 2.3.1, the Operate phase is describe in 2.3.3.  The structure of A-Plans and D-Prognoses are detailed in 2.4.2 and 2.4.3. |

##### Create Flex Offers

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Plan, Validate | NA | Create flexibility offers | |
| **Mapping name** | AGR\_FLEX\_OFFER\_DETERMINE\_FLEXIBILITY |
| **Context** | An Aggregator creates and delivers flex offers to participants (DSOs and BRPs) that have sent flex requests.  This Pluggable Business Component is called per role (BRP or DSO) and analyses the flex requests and checks if it is able to supply any flexibility for the congestion point or connections and the PTU(s) it applies to. If so, a list of flex offers detailing the flexibility that can be supplied and the price setting for doing so is returned.  The input for this PBC is a list of flex requests, prognoses from either a DSO or a BRP, the portfolio, a list of connection groups and associated connections and a list of previously created flex offers. The output is a list of flex offers for the DSO or BRP. Note that the result list may be empty, if no flex is available. |
| **Input** | * int PTU\_DURATION   The duration of a PTU expressed in minutes.   * org.joda.time.LocalDate PERIOD   The period for which flex offers   * List <energy.usef.agr.dto.ConnectionPortoflioDTO> CONNECTION\_PORTFOLIO\_DTO   The connection portfolio.   * Map <String, List <String>> CONNECTION\_GROUPS\_TO\_CONNECTIONS\_MAP   Connections related to the connection groups for the given period   * List <energy.usef.core.workflow.dto.FlexRequestDto> FLEX\_REQUEST\_DTO\_LIST   The list of flex requests for which flex offers can be created.   * List <energy.usef.core.workflow.dto.PrognosisDto> LATEST\_A\_PLANS\_DTO\_LIST   The list of latest A-plans.   * List <energy.usef.core.workflow.dto.PrognosisDto> LATEST\_D\_PROGNOSES\_DTO\_LIST   The list of latest D-prognoses.   * List <energy.usef.core.workflow.dto.FlexOfferDto> FLEX\_OFFER\_DTO\_LIST   The list of already placed flex offers. |
| **Output** | * List <energy.usef.core.workflow.dto.FlexOfferDto> FLEX\_OFFER\_DTO\_LIST   The list of flex offers that need to be sent. Each flex offer contains a list of PTUs with the determined flexibility. |
| **References** | See [2]: chapter 5.2.12 AgrFlexOfferCoordinator  See [1]: AGR flexibility offers are introduced in section 2.3.1.3 and 2.3.2.1, with the exchange process being described in 2.3.2.4 and the structure in 2.4.4. |

##### Initialize Non-UDI Clusters

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Operate | NA |  | |
| **Mapping name** | AGR\_INITIALIZE\_NON\_UDI\_CLUSTERS |
| **Context** | The Non-UDI aggregator wants share the common reference information for a certain period with his aggregator-in-a-box (demand-response) solution.  Common reference information is retrieved for the period specified in the triggering event. All connection groups with connections that are active on the specified day are retrieved and will be sent to the aggregator-in-a-box (demand-response) solution.  This Pluggable Business Component sends the common reference information to the aggregator-in-a-box (demand-response) solution.  This PBC receives as input the period it is invoked for, the entity addresses of all Connections represented by the AGR during that period, including the entity address of the BRP for each Connection and the entity addresses of any congestion points, plus a list of Connections belonging to each of those points |
| **Input** | * org.joda.time.LocalDate PERIOD   The period for which the cluster needs to be initialized.   * Map (String, List(String)) BRP\_CONNECTION\_LIST\_MAP   Map with BRPs and for each BRP the list of connection entity addresses.   * Map (String, List(String)) CP\_CONNECTION\_LIST\_MAP   Map with congestion point identifications and for each congestion point the list of connection entity addresses. |
| **Output** | none |
| **References** | See [2]: chapter 5.4.1 AgrNonUdiInitializeCoordinator |

##### Retrieve ADS Goals

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Operate | NA |  | |
| **Mapping name** | AGR\_NON\_UDI\_RETRIEVE\_ADS\_GOAL\_REALIZATION |
| **Context** | Non-UDI aggregators have fairly limited control over the execution of their plans. They will have to rely on their external ADS solution to execute these to the best of its ability. However, Non-UDI aggregators monitor their ADS and feed its statistics about goal realization into the Detect Deviations from A-Plans and/or D-Prognoses process so plans can be adjusted, if necessary.  This Pluggable Business Component retrieves the power values for all connection groups (BRPs and congestion points) from the external ADS system.  This PBC receives as input the period it is invoked for, the PTU duration and the current portfolio and returns the updated portfolio with added observed power values. |
| **Input** | * org.joda.time.LocalDate PERIOD   The period for which the power data is retrieved from the external ADS solution.   * int PTU\_DURATION   The duration of a PTU expressed in minutes.   * List <energy.usef.agr.dto.ConnectionGroupPortfolioDTO> CURRENT\_PORTFOLIO:   The current portfolio. |
| **Output** | * List <energy.usef.agr.dto.ConnectionGroupPortfolioDTO> UPDATED\_PORTFOLIO:   The current portfolio. |
| **References** | See [2]: chapter 5.4.2 AgrNonUdiRetrieveAdsGoalRealizationCoordinator |

##### Determine Net Demands via ADS

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Operate | NA | Determine Net Demands | |
| **Mapping name** | AGR\_DETERMINE\_NET\_DEMANDS |
| **Context** | An Aggregator needs to fetch detailed net monitoring information of the devices behind each connection represented via the ADS. The information is used to detect deviations from A-plan and/or D-prognoses  This Pluggable Business Component fetches the net demand information for each connection in a connection group that is managed by the Aggregator and returns the connections with added UDIs, UDI events and the current DTUs |
| **Input** | * int PTU\_DURATION   The duration of a PTU expressed in minutes.   * List <energy.usef.agr.dto.ConnectionPortfolioDto> CONNECTION\_PORTFOLIO\_DTO\_LIST   The complete portfolio. |
| **Output** | * List <energy.usef.agr.dto.ConnectionPortfolioDto> CONNECTION\_PORTFOLIO\_DTO\_LIST   The list with all the connections with added UDIs, UDI events and the current DTUs.   * List< energy.usef.agr.dto.UdiEventDto> UPDATED\_UDI\_EVENT\_DTO\_LIST   List of changed UDI events and associated capabilities. |
| **References** | See [2]: chapter 5.4.3 AgrDetermineNetDemandCoordinator  See [1]: The process flow in the Operate phase is described in section 2.3.3. |

##### Identify Change In Forecast

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Operate | NA | Identify changes in forecast | |
| **Mapping name** | AGR\_IDENTIFY\_CHANGE\_IN\_FORECAST |
| **Context** | An aggregator needs to identify relevant changes in the forecast during the Plan phase, which may impact its portfolio and the A-plan(s) and D-prognoses previously sent to other participants. Also, in case the forecast has actually changed, I need to trigger my portfolio re-optimization.  This Pluggable Business Component determines the forecast changes based on the current connection portfolio makes the decision if the connection portfolio needs to be re-optimized. |
| **Input** | * org.joda.time.LocalDate PERIOD:   The period for which changes in the forecast are identified   * int PTU\_DURATION   The duration of a PTU expressed in minutes.   * List <energy.usef.agr.dto.ConnectionPortfolioDTO> CONNECTION\_PORTFOLIO:   The connection portfolio for the specified period. |
| **Output** | * boolean FORECAST\_CHANGED:   Boolean value indicating whether a significant change in the forecast was identified.   * List < energy.usef.core.dto.PtuContainerDto> FORECAST\_CHANGED\_PTU\_CONTAINER\_DTO\_LIST:   List of PTU containers that for which a change in forecast has been identified. |
| **References** | See [2]: chapter 5.4.4 AgrIdentifyChangeInForecastCoordinator  See [1]: The process flow in the Plan phase is described in section 2.3.1.  The structure of A-Plans is detailed in 2.4.2. |

##### Detect Prognoses Deviations

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Operate | NA | Detect deviations from Prognoses | |
| **Mapping name** | AGR\_DETECT\_DEVIATION\_FROM\_PROGNOSES |
| **Context** | An Aggregator needs to compare the actual and forecast usage of its connections with the current D-prognoses and A-plans to make sure that all contractual obligations can be met. If not, the Aggregator’s portfolio needs to be re-optimized.  This Pluggable Business Component detects deviations between the calculated forecast and prognoses per PTU and returns the PTU indexes for which deviations are detected.  The input for this PBC is the period, the PTU duration, the current PTU index, the forecast for all connections in a connection group and the latest prognosis for that connection group on the given day. The output is an array identifying PTU indexes where deviation is detected. |
| **Input** | * org.joda.time.LocalDate PERIOD:   The period for which changes in the forecast are identified   * int PTU\_DURATION   PTU duration.   * int CURRENT\_PTU\_INDEX   Current PTU index.   * energy.usef.agr.dto.ConnectionGroupPortfolioDTO CONNECTION\_PORTFOLIO:   The connection portfolio for the given connection group and given day.   * energy.usef.core.workflow.dto.PrognosisDto LATEST\_PROGNOSIS   All latest A-plan or D-prognoses for the given connection group and given day. |
| **Output** | * List <Integer> DEVIATION\_INDEX\_LIST   A list with PTU indexes where deviation is detected. |
| **References** | See [2]: chapter 5.4.5 AgrDetectDeviationCoordinator  See [1]: The process flow in the Operate phase is described in section 2.3.3. |

##### Non-UDI Detect Prognoses Deviations

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Operate | NA | Detect deviations from Prognoses | |
| **Mapping name** | AGR\_NON\_UDI\_DETECT\_DEVIATION\_FROM\_PROGNOSES |
| **Context** | A Non-UDI Aggregator needs to compare the actual and forecast usage of its connections with the current D-prognoses and A-plans to make sure that all contractual obligations can be met. If not, the Aggregator’s portfolio needs to be re-optimized.  This Pluggable Business Component detects deviations between the calculated forecast and prognoses per PTU and returns the PTU indexes for which deviations are detected.  The input for this PBC is the period, the PTU duration, the current PTU index, the forecast for all connections in a connection group and the latest prognoses for that connection group on the given day. The output is an array identifying PTU indexes where deviation is detected. |
| **Input** | * org.joda.time.LocalDate PERIOD:   The period for which changes in the forecast are identified   * int PTU\_DURATION   PTU duration.   * int CURRENT\_PTU\_INDEX   Current PTU index.   * energy.usef.agr.dto.ConnectionGroupPortfolioDTO CONNECTION\_PORTFOLIO:   The connection portfolio for the given connection group and given day.   * energy.usef.core.workflow.dto.PrognosisDto LATEST\_PROGNOSIS   The latest A-plan or D-prognosis for the given connection group and given day. |
| **Output** | * List <Integer> DEVIATION\_INDEX\_LIST   A list with PTU indexes where deviation is detected. |
| **References** | See [2]: chapter 5.4.5 AgrDetectDeviationCoordinator  See [1]: The process flow in the Operate phase is described in section 2.3.3. |

##### Send ADS Messages

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Operate | NA | Realize A-plan and/or D-Prognoses by controlling ADS | |
| **Mapping name** | AGR\_CONTROL\_ACTIVE\_DEMAND\_SUPPLY |
| **Context** | After portfolio optimization a UDI Aggregator sends device messages to ADS endpoints to steer consumption and/or production on the connections it represents.  This Pluggable Business Component sends a device message to the applicable ADS endpoint and returns any failed messages, if any. |
| **Input** | * energy.usef.agr.dto.device.request.DeviceMessageDto DEVICE\_MESSAGE\_DTO   The UDI Control Message (device message). |
| **Output** | * energy.usef.agr.dto.device.request.DeviceMessageDto FAILED\_DEVICE\_MESSAGE\_DTO   A UDI Control Message (device message) that could not be processed or sent (if any). |
| **References** | See [2]: chapter 5.4.6 AgrControlActiveDemandSupplyCoordinator  See [1]: The process flow in the Operate phase is described in sections 2.3.3.  See [4]: UDI (the mechanism used by the AGR to communicate with Prosumers) is described in section 4.4. |

##### Send ADS Goals

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Operate | NA |  | |
| **Mapping name** | AGR\_NON\_UDI\_SET\_ADS\_GOALS |
| **Context** | A Non-UDI aggregator communicates detailed ADS goals to his "aggregator in a box" solution. This is done every time a new A-plan or D-prognosis is created.  This Pluggable Business Component transforms the prognosis from the input into the applicable format and sends it to his “aggregator-in-a-box” solution. |
| **Input** | * org.joda.time.LocalDate PERIOD   The period for which the ADS goals are sent to the external ADS solution.   * int PTU\_DURATION   The duration of a PTU expressed in minutes.   * energy.usef.core.workflow.dto.PrognosisDto PROGNOSIS\_DTO   The A-plan or D-prognosis that needs to be sent to the "aggregator in a box" solution. |
| **Output** | None |
| **References** | See [2]: chapter 5.4.7 AgrNonUdiSetAdsGoalsCoordinator |

##### Initiate Settlement

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | | AGR | Settlement | NA | Validate if A-plans/D-prognoses Were Met | | |
| **Mapping name** | AGR\_INITIATE\_SETTLEMENT |
| **Context** | An Aggregator validates and processes the settlement messages sent by the DSO and BRP. These settlement messages contain details of purchased flex and penalties for non-delivery of such, and serve as the basis for invoices to the DSOs and BRPs.  The Aggregator prepares the settlement before the settlement message is received and processed.  This Pluggable Business Component prepares the settlement for the specified period based on the provided prognoses, flex requests, offers and orders and the portfolio. It returns a settlement object that contains all settlement items for the supplied flex orders. |
| **Input** | * org.joda.time.LocalDate START\_DATE   Start date of the period to initiate settlement for.   * org.joda.time.LocalDate END\_DATE   End date of the period to initiate settlement for.   * List< energy.usef.core.workflow.dto.PrognosisDto> PROGNOSIS\_DTO\_LIST   The list of all relevant prognoses in the specified period.   * List< energy.usef.core.workflow.dto.FlexRequestDto> FLEX\_REQUEST\_DTO\_LIST   List of all relevant flex requests in the specified period.   * List <energy.usef.core.workflow.dto.FlexOfferDto> FLEX\_OFFER\_DTO\_LIST   List of all relevant flex offers in the specified period.   * List <energy.usef.core.workflow.dto.FlexOrderDto> FLEX\_ORDER\_DTO\_LIST   List of all relevant flex orders in the specified period.   * List <energy.usef.agr.dto.ConnectionDto> CONNECTION\_PORTFOLIO\_DTO\_LIST   The connection portfolio for the specified period   * Map<LocalDate, List<energy.usef.agr.dto.ConnectionGroupPortfolioDto>> CONNECTION\_GROUP\_PORTFOLIO\_DTO\_PER\_PERIOD\_MAP   Map with the portfolio on connection group level per period   * Map <String, List <String>> CONNECTION\_GROUPS\_TO\_CONNECTIONS\_MAP   Map with connection group identifiers and associated connection identifiers.   * Map <String, energy.usef.core.workflow.dto.USEFRoleDto> CONNECTION\_GROUPS\_TO\_USEF\_ROLE\_MAP:   Association of the USEF identifier of a connection group and its type (DSO or BRP).   * int PTU\_DURATION   The duration of a PTU expressed in minutes. |
| **Output** | * energy.usef.core.workflow.dto.SettlementDto SETTLEMENT\_DTO The settlement object that contains a list of all settlement items for the supplied flex orders. |
| **References** | See [2]: chapter 5.5.1 AgrInitiateSettlementCoordinator  See [1]: The Settlement processes between Aggregator and BRP and DSO are elaborated in section 2.3.4. |

##### Validate Settlement Items

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | AGR | Settlement | NA | Validate BRP/DSO Settlement Items | |
| **Mapping name** | AGR\_VALIDATE\_SETTLEMENT\_ITEMS |
| **Context** | An Aggregator validates and processes the settlement messages sent by the DSO and BRP. These settlement messages contain details of purchased flex and fines for non-delivery of such, and serve as the basis for invoices to the DSOs and BRPs.  The Aggregator prepares the settlement before the settlement message is processed.  This Pluggable Business Component compares the settlement message with the preparation performed earlier and compares the amount of flex sold and actually delivered for all settlement items in the message. It returns whether the settlement message is accepted or disputed.  The input for this PBC is the reference of the flex order related to this settlement, the settlement items received from the DSO or BRP and the list of settlement items the aggregator prepared earlier.  The output is the result of the analysis, i.e. whether the settlement of the DSO or BRP has been accepted or disputed. Acceptable values are: ACCEPTED and DISPUTED. |
| **Input** | * String ORDER\_REFERENCE  Reference of the flex order which concludes the flex exchange to settle. * String COUNTER\_PARTY\_ROLERole of the party from which the settlement was received. * energy.usef.core.workflow.dto.FlexOrderSettlementDto   RECEIVED\_FLEX\_ORDER\_SETTLEMENT The settlement received from the counter party which contains the data needed to finalize the settlement.   * List < energy.usef.core.workflow.dto.FlexOrderSettlementDto> PREPARED\_FLEX\_ORDER\_SETTLEMENTS   List of prepared flex order settlements on a given date. |
| **Output** | * energy.usef.core.workflow.dto.DispositionAcceptedDisputedDto FLEX\_ORDER\_SETTLEMENT\_DISPOSITION The result of the PBC, whether the settlement of the DSO has been accepted or disputed. Acceptable values are:   + ACCEPTED   + DISPUTED |
| **References** | See [2]: chapter 5.5.2 AgrReceiveSettlementMessageCoordinator  See [1]: The Settlement processes between Aggregator and BRP and DSO are elaborated in section 2.3.4. |

#### Balance Responsible Party

##### Received A-plan

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | BRP | Plan | Green | Assess A-Plan | |
| **Mapping name** | BRP\_RECEIVED\_APLAN |
| **Context** | A BRP wants to optimize its portfolio and attain an economically optimized program.  It assesses the A-plans received from Aggregators for a certain period and creates flex requests for these Aggregators if applicable.  This Pluggable Business Component validates the supplied A-plans and returns a list of accepted A-plans and a list of processed A-plans for which flex trading is supposed to take place. |
| **Input** | * List <energy.usef.core.workflow.dto.PrognosisDto> A\_PLAN\_DTO\_LIST   List of all A-plans for a certain period with the prognosis data from the planboard, for the related connections, day and PTU indexes.   * List <energy.usef.core.workflow.dto.PrognosisDto> RECEIVED\_A\_PLAN\_DTO\_LIST   List if all A-plans with status received.   * int PTU\_DURATION The duration of a PTU expressed in minutes. |
| **Output** | * List <energy.usef.core.workflow.dto.PrognosisDto> ACCEPTED\_A\_PLAN\_DTO\_LIST   List of A-Plans that are accepted.   * List <energy.usef.core.workflow.dto.PrognosisDto> PROCESSED\_A\_PLAN\_DTO\_LIST   List of A-plans for which flex trading is supposed to take place. |
| **References** | See [2]: chapter 5.2.16 BrpAplanCoordinator  See [1]: The process of exchanging A-plans from Aggregator to BRP is described in sections 2.3.1.2. Section 2.3.1.3 describes the process of flexibility trading between Aggregator and BRP. A-plans are described in section 2.4.2, section 2.4.4.1 describes the mechanics of flexibility requests |

##### Prepare Flex Requests

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | BRP | Plan | Yellow | Create flexibility requests | |
| **Mapping name** | BRP\_PREPARE\_FLEX\_REQUESTS |
| **Context** | A BRP wants to optimize its portfolio and attain an economically optimized program.  It assesses the A-plans received from Aggregators and creates flex requests for these Aggregators if applicable.  This Pluggable Business Component validates the supplied processed A-plans and generates flex requests. |
| **Input** | * List <energy.usef.core.workflow.dto.PrognosisDto> PROCESSED\_A\_PLAN\_DTO\_LIST   List of A-plans, one per Aggregator, for which flex trading should take place.   * int PTU\_DURATION The duration of a PTU expressed in minutes. |
| **Output** | * List <energy.usef.core.workflow.dto.FlexRequestDto> FLEX\_REQUEST\_DTO\_LIST   A list with the different flex requests to be sent to the Aggregators.   * List <energy.usef.core.workflow.dto.PrognosisDto> ACCEPTED\_A\_PLAN\_DTO\_LIST   List of A-plans that have changed from status processed to status accepted. |
| **References** | See [2]: chapter 5.2.16 BrpAplanCoordinator  See [1]: The process of exchanging A-plans from Aggregator to BRP is described in sections 2.3.1.2. Section 2.3.1.3 describes the process of flexibility trading between Aggregator and BRP. A-plans are described in section 2.4.2, section 2.4.4.1 describes the mechanics of flexibility requests. |

##### Get Not Desirable Flex Offers

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | BRP | Plan | Green | Create Flex Orders | |
| **Mapping name** | BRP\_GET\_NOT\_DESIRABLE\_FLEX\_OFFERS |
| **Context** | When optimizing its portfolio, a BRP creates orders for the flexibility offered earlier by Aggregators.  This Pluggable Business Component decides which of the supplied flex offers are desirable and should possibly be turned into flex orders. It returns a list of sequence numbers of the flex offers which are not desirable.  The input for this PBC is the identification of the connection group, and a list of planboard messages representing the flex offers for the specified connection group, with per flex offer the offer per PTU. The output is a list of sequence numbers of the flex offers which are not desirable and will not turn into a flex order. |
| **Input** | The PBC is called for every connection group identified by the BRP.   * String CONNECTION\_GROUP\_IDENTIFIER   Identification of the connection group.   * List <energy.usef.core.workflow.dto.FlexOfferDto> FLEX\_OFFER\_DTO\_LIST   A list of flex offers for the specified connection group, with per flex offer the offer per PTU. |
| **Output** | * List <Long> NOT\_DESIRABLE\_FLEX\_OFFER\_SEQUENCE\_LIST   List of sequence numbers of the flex offers, which are not desired and will not turn into flex orders. |
| **References** | See [2]: chapter 5.2.18 BrpFlexOrderCoordinator  See [1]: BRP flexibility orders are described in section 2.3.1.3, with the structure being described in 2.4.4. |

##### Create Flex Orders

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | BRP | Plan/Operate | Green | Create Flex Orders | |
| **Mapping name** | BRP\_PLACE\_FLEX\_ORDERS |
| **Context** | When optimizing its portfolio, a BRP creates orders for the flexibility offered earlier by Aggregators.  This Pluggable Business Component decides which of the supplied and desired flex offers should be turned into flex orders. It returns a list of sequence numbers of the flex offers which are accepted and will turn into flex orders.  The input for this PBC is the identification of the connection group and a list of planboard messages representing the desired flex offers for the specified connection group, with per flex offer the offer per PTU. The output is a list of sequence numbers of the flex offers which are accepted and will turn into a flex order. |
| **Input** | * String CONNECTION\_GROUP\_IDENTIFIER   Identification of the connection group.   * List <energy.usef.core.workflow.dto.FlexOfferDto> FLEX\_OFFER\_DTO\_LIST   A list of desired flex offers for the specified connection group, with per flex offer the offer per PTU in the specified period.   * int PTU\_DURATION   The duration of a PTU expressed in minutes.   * org.joda.time.LocalDate PERIOD   The day for which flex orders will be sent. |
| **Output** | * List <Long> ACCEPTED\_FLEX\_OFFER\_SEQUENCE\_LIST   List of sequence numbers of the flex offers which are accepted and will turn into flex orders. |
| **References** | See [2]: chapter 5.2.18 BrpFlexOrderCoordinator  See [1]: BRP flexibility orders are described in section 2.3.1.3, with the structure being described in 2.4.4. |

##### Create Missing A-Plans

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | BRP | Plan | NA | Approve A-plan | |
| **Mapping name** | BRP\_CREATE\_MISSING\_APLANS |
| **Context** | At day ahead gate closure, the BRP generates any missing A-plans for aggregators that did not send an A-plan.  This Pluggable Business Component generates an A-plan, given the period, ptu duration, connection count and Internet domain for the aggregator as input. The result looks just like a normal A-plan, except that the SenderDomain is the BRP domain. |
| **Input** | * org.joda.time.LocalDate PERIOD   The day for which the missing A-plan needs to be created.   * int PTU\_DURATION   The duration of a PTU expressed in minutes.   * int CONNECTION\_COUNT   Number of connections in the missing A-plan.   * String AGGREGATOR\_DOMAIN   USEF identifier of the aggregator for which an A-plan needs to be created. |
| **Output** | * energy.usef.core.workflow.dto.PrognosisDto PROGNOSIS\_DTO   The created missing A-plan |
| **References** | See [2]: chapter 5.2.21 BrpCreateMissingAPlansCoordinator |

##### Initiate Settlement

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | BRP | Settlement | NA | Validate if A-plans Were Met | |
| **Mapping name** | BRP\_INITIATE\_SETTLEMENT |
| **Context** | At the start of the settlement process a BRP generates settlement items based on the flex orders of previous month and the actual power consumption on the related connections. For each settlement item, the delivered flex quantity and power deficiency for the PTU is calculated as well as a possible penalty.  This Pluggable Business Component prepares the settlement for the specified period based on the provided prognoses, flex requests, offers and orders and the metered data. It returns a settlement object that contains all settlement items for the supplied flex orders. |
| **Input** | * org.joda.time.LocalDate START\_DATE   Start date of the period to initiate settlement for.   * org.joda.time.LocalDate END\_DATE   End date of the period to initiate settlement for.   * List< energy.usef.core.workflow.dto.PrognosisDto> PROGNOSIS\_DTO\_LIST   The list of all relevant prognoses in the specified period.   * List< energy.usef.core.workflow.dto.FlexRequestDto> FLEX\_REQUEST\_DTO\_LIST   List of all relevant flex requests in the specified period.   * List <energy.usef.core.workflow.dto.FlexOfferDto> FLEX\_OFFER\_DTO\_LIST   List of all relevant flex offers in the specified period.   * List <energy.usef.core.workflow.dto.FlexOrderDto> FLEX\_ORDER\_DTO\_LIST   List of all relevant flex orders in the specified period.   * List <energy.usef.core.data.xml.bean.message.MeterDataSet> SMART\_METER\_DATA   The metered data for the specified period |
| **Output** | * energy.usef.core.workflow.dto.SettlementDto SETTLEMENT\_DTO The settlement object that contains a list of all settlement items for the supplied flex orders. |
| **References** | See [2]: chapter 5.5.3 BrpInitiateSettlementCoordinator  See [1]: The Settlement process between BRP and Aggregator is elaborated in sections 2.3.4 |

##### Calculate Penalty Amount

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | BRP | Settlement | NA | Calculate Flex Prices and Penalty Amounts | |
| **Mapping name** | BRP\_REQUEST\_PENALTY\_DATA |
| **Context** | At the start of the settlement process a BRP generates settlement items based on the previous month’s flex orders and the actual power consumption on the related connections. For each settlement item, the delivered flex quantity and power deficiency for the PTU is calculated as well as a possible penalty.  This Pluggable Business Component calculates the penalty for all provided settlement items for the specified Aggregator and returns them.  The input for this PBC is the settlement object and the PTU duration. The output is the input settlement object updated with penalty data. |
| **Input** | * energy.usef.core.workflow.dto.SettlementDto SETTLEMENT\_DTO The settlement object that contains a list of all settlement items for the supplied flex orders. * int PTU\_DURATION   The duration of a PTU expressed in minutes |
| **Output** | * energy.usef.core.workflow.dto.SettlementDto SETTLEMENT\_DTO   The input settlement object updated with penalty data. |
| **References** | See [2]: chapter 5.5.3 BrpInitiateSettlementCoordinator  See [1]: The Settlement process between BRP and Aggregator is elaborated in sections 2.3.4 |

#### Distribution System Operator

##### Create Non Aggregator Forecast

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | DSO | Plan | NA | Create Non Aggregator Forecast | |
| **Mapping name** | DSO\_CREATE\_NON\_AGGREGATOR\_FORECAST |
| **Context** | When the DSO initializes its planboard it needs to create and store the Non-Aggregator connection forecast information.  This Pluggable Business Component generates a load forecast for all connections within the specified congestion point which are not served by an Aggregator. The result is the total forecast for all PTUs in a day for the connections not served by an Aggregator. |
| **Input** | * String CONGESTION\_POINT\_ENTITY\_ADDRESS   The entity address associated with the congestion point.   * String PTU\_DATE   The day for which the forecast is to be created in format yyyy-MM-dd.   * int PTU\_DURATION   The duration of a PTU expressed in minutes   * List <String> AGR\_DOMAIN\_LIST   A list of aggregator domains associated to the congestion point.   * List <long> AGR\_CONNECTION\_COUNT\_LIST   The number of connections served by each Aggregator plus one entry representing the number of connections not served by an aggregator. |
| **Output** | * List <long> POWER   Power forecast expressed in Watt.   * List <long> MAXLOAD   Maximum load forecast expressed in Watt for the day.  The two lists must have the same size and should in total specify the forecast for all PTUs in a day. |
| **References** | See [2]: chapter 5.2.24 DsoConnectionForecastPlanBoardCoordinator  See [1]: The need and the usage of non-Aggregator forecasts is elaborated in section 4.3.2.1. |

##### Create Missing D-Prognoses

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | DSO | Validate | NA | Supply Missing D-Prognoses | |
| **Mapping name** | DSO\_CREATE\_MISSING\_DPROGNOSES |
| **Context** | To perform a proper grid safety analysis for a congestion point, the DSO needs valid D-prognoses for all aggregators. If not all aggregators have delivered a valid D-prognosis at the day-ahead gate closure time, the DSO has an incomplete forecast for a congestion point.  This Pluggable Business Component generates the missing D-prognosis for the specified congestion point, date and aggregator. |
| **Input** | * String CONGESTION\_POINT\_ENTITY\_ADDRESS   The entity address associated with the congestion point for which a D-prognosis should be created   * String AGGREGATOR\_DOMAIN   The aggregator domain for which a D-prognosis should be created   * org.joda.time.LocalDate ANALYSIS\_DAY   The day for which a D-prognosis should be created   * Integer PTU\_DURATION   The PTU duration   * Integer AGGREGATOR\_CONNECTION\_AMOUNT   The amount of connections the aggregator has within this congestion point. |
| **Output** | * energy.usef.core.workflow.dto.PrognosisDto D\_PROGNOSIS   The generated D-prognosis for the specified congestion point |
| **References** | See [2]: chapter 5.3.2 DsoCreateMissingDPrognosisCoordinator  See [1]: DSO grid safety analysis is introduced as part of the Validate phase in section 2.3.2. It is described as the trigger for flexibility request messages (section 2.4.4). |

##### Perform Grid Safety Analysis

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | DSO | Validate | NA | Create Grid Safety Analysis | |
| **Mapping name** | DSO\_CREATE\_GRID\_SAFETY\_ANALYSIS |
| **Context** | A DSO wants to identify whether congestion is expected or not. After that, the DSO can decide whether the congestion can be solved by purchasing flexibility or not.  This Pluggable Business Component performs the analysis for a certain congestion point based on D-prognoses received from Aggregators and Non-Aggregator connection forecasts for those connections not related to an Aggregator. The resulting Grid Safety Analysis is then used to initiate flex trading.  The input for this PBC is the non Aggregator connection forecast information for a congestion point and a list of D-prognosis of each Aggregator related to the congestion point. The output is the result of the Grid Safety Analysis. |
| **Input** | * String PERIOD   The day for which the grid safety analysis should be executed.   * String CONGESTION\_POINT\_ENTITY\_ADDRESS   The congestion point for which the grid safety analysis will be executed.   * List <energy.usef.core.workflow.dto.PrognosisDTO> D\_PROGNOSIS\_LIST   List of D-prognoses related to the congestion point.   * energy.usef.core.workflow.dto.NonAggregatorForecastDto NON\_AGGREGATOR\_FORECAST   The non-aggregator forecast related to the congestion point. |
| **Output** | * energy.usef.dso.workflow.dto.GridSafetyAnalysisDto GRID\_SAFETY\_ANALYSIS   The result of the Grid Safety Analysis. |
| **References** | See [2]: chapter 5.3.3 DsoGridSafetyAnalysisCoordinator  See [1]: DSO grid safety analysis is introduced as part of the Validate phase in section 2.3.2. It is described as the trigger for flexibility request messages (section 2.4.4). |

##### Post Coloring Process

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | DSO | Validate | NA | Prepare stepwise connection limiting & recovery | |
| **Mapping name** | DSO\_POST\_COLORING\_PROCESS |
| **Context** | If the grid safety analysis indicates that congestion is expected in a certain PTU for a certain congestion point, but there no flexibility can be procured to resolve the expected congestion, PTU(s) need to be marked as Orange in order to prepare for connection limiting.  This Pluggable Business Component takes further action, once the coloring process is done. |
| **Input** | None |
| **Output** | None |
| **References** | See [2]: chapter 5.3.4 DsoColoringProcessCoordinator  See [1]: DSO grid safety analysis and identification of grid constraints in the Validate phase is described in section 2.3.2.3.1. |

##### Prepare Stepwise Limiting

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | DSO | Validate | NA | Prepare Stepwise Conection Limiting & Recovery | |
| **Mapping name** | DSO\_PREPARE\_STEPWISE\_LIMITING |
| **Context** | If, in the validate phase, the grid safety analysis or the flex ordering process indicates that congestion is expected that cannot be solved by flex ordering, each PTU for which congestion is expected is colored Orange on the planboard. Also, a plan is created for limiting the connections in these PTUs, in order to perform stepwise connection limiting in the Operate phase.  This Pluggable Business Component creates this plan based on the Congestion Point and the applicable PTU list. |
| **Input** | * String CONGESTION\_POINT\_ENTITY\_ADDRESS   The entity address associated with the congestion point for which congestion is expected.   * List <energy.usef.core.workflow.dto.PtuContainerDto> PTU\_CONTAINERS   PTUs for which congestion is expected. |
| **Output** | None |
| **References** | See [2]: chapter 5.3.4 DsoColoringProcessCoordinator  See [1]: DSO grid safety analysis and identification of grid constraints in the Validate phase is described in section 4.3.2. |

##### Create Flex Requests

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | DSO | Validate | Yellow | Create Flex Requests | |
| **Mapping name** | DSO\_CREATE\_FLEX\_REQUEST |
| **Context** | When congestion is expected for a certain congestion point during a certain PTU, the DSO creates and sends flex requests to all Aggregators linked to an identified congestion point.  This Pluggable Business Component creates flex requests based on the specified congestion point, PTUs and grid safety analysis. The result is a list of flex requests for all Aggregators related to the specified congestion point.  The input for this PBC is the entity address associated with the congestion point, the day for which flex request will be sent, and the data of the Grid Safety Analysis, for the related congestion point, day and PTU indexes. The output is a list of flex requests to be sent to the Aggregators related to the congestion point. |
| **Input** | * String CONGESTION\_POINT\_ENTITY\_ADDRESS   The entity address associated with the congestion point.   * Date PERIOD   The day for which flex request will be sent.   * energy.usef.dso.workflow.dto.GridSafetyAnalysisDto GRID\_SAFETY\_ANALYSIS\_LIST   The grid safety analysis for the related congestion point, day and PTU indexes. |
| **Output** | * List <energy.usef.core.workflow.dto.FlexRequestDto> FLEX\_REQUESTS\_DTO\_LIST   List with the different flex requests to be sent to the Aggregators related to the congestion point. |
| **References** | See [2]: chapter 5.3.5 DsoCreateFlexRequestCoordinator  See [1]: DSO flexibility requests are first mentioned in section 2.3.2.3, where the requirement to wait for all D-prognoses from AGRs is specified. Section 2.3.2.4 is the main section describing the role of DSO flexibility requests, and section 2.4.4.1 describes the mechanics. |

##### Create Flex Orders

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | DSO | Validate | Yellow | Create Flex Orders | |
| **Mapping name** | DSO\_PLACE\_FLEX\_ORDERS |
| **Context** | To avoid congestion, a DSO creates orders for the flexibility offered earlier by Aggregators.  This Pluggable Business Component decides which of the supplied flex offers should be turned into flex orders based. It returns a list of flex offers which are accepted and will turn into flex orders.  The input for this PBC is the EAN code identifying the congestion point and a list of flex offers for the specified congestion point, with per flex offer the offer per PTU. |
| **Input** | * List< energy.usef.core.workflow.dtoFlexOfferDto> FLEX\_OFFER\_DTO\_LIST   List of Flex offers containing all flex offers for the associated congestion point in the specified period.   * String CONGESTION\_POINT\_ENTITY\_ADDRESS   Identification of the congestion point.   * int PTU\_DURATION   The duration of a PTU expressed in minutes   * Date PERIOD   The day for which flex orders will be sent. |
| **Output** | * List <Long> ACCEPTED\_FLEX\_OFFER\_SEQUENCE\_LIST   List of sequence numbers of the flex offers which are accepted and will turn into flex orders. |
| **References** | See [2]: chapter 5.3.6 DsoFlexOrderCoordinator  See [1]: DSO flexibility orders are introduced in section 2.3.2.1, with the exchange process being described in 2.3.2.4 and the structure in 2.4.4. |

##### Monitor Grid

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | DSO | Operate | NA | Monitor Grid | |
| **Mapping name** | DSO\_MONITOR\_GRID |
| **Context** | The DSO monitors the grid in the operate phase in order to identify and resolve possible congestion on the grid.  This Pluggable Business Component monitors based the congestion point, the amount of limited power at this PTU, and the number of connections and returns the actual and maximum load values per-PTU and indicates if congestion is detected.  The input for this PBC is the EAN code identifying the congestion point, the amount of limited power at this PTU, and the number of connections. The output is the actual load (W) at this moment, the maximum load (W) at this moment, and the CONGESTION Boolean (true if congestion occurred, false otherwise). |
| **Input** | * String CONGESTION\_POINT\_ENTITY\_ADDRESS   The entity address associated with the congestion point.   * Long LIMITED\_POWER   The amount of limited power at this PTU   * Long NUM\_CONNECTIONS   Number of connections |
| **Output** | * Long ACTUAL\_LOAD   Actual load (W) at this moment   * Long MAX\_LOAD   Maximum load (W) at this moment   * Long MIN\_LOAD   Minimum load (W) at this moment   * Boolean CONGESTION   true if congestion occurred, false otherwise |
| **References** | See [2]: chapter 5.4.8 DsoOperateCoordinator  See [1]: The need and the usage of DSO grid monitoring in the operate phase is elaborated in sections 2.3.3.4 and 4.3.3.1. |

##### Place Operate Flex Orders

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | DSO | Operate | NA | Place Flexibility Orders | |
| **Mapping name** | DSO\_PLACE\_OPERATE\_FLEX\_OPDERS |
| **Context** | In case congestion is detected by the DSO, the DSO verifies if there are any valid flexibility offers available and, if possible, places flex orders.  This Pluggable Business Component decides which of the supplied flex offers should be turned into flex orders based on the supplied grid safety analysis. It returns a list of flex offers which are accepted and will turn into flex orders. |
| **Input** | * List< energy.usef.core.workflow.dtoFlexOfferDto> FLEX\_OFFER\_DTO\_LIST   List of Flex offers containing all flex offers for the associated congestion point from the current day onwards.   * energy.usef.dso.workflow.dto;GridSafetyAnalysisDto GRID\_SAFETY\_ANALYSIS\_DTO   The grid safety analysis |
| **Output** | * List< energy.usef.core.workflow.dtoFlexOfferDto> ACCEPTED\_FLEX\_OFFER\_DTO\_LIST   A list of flex offers which are accepted and will turn into flex orders. |
| **References** | See [2]: chapter 5.4.8 DsoOperateCoordinator  See [1]: When congestion is detected in the operate phase, this situation may be solved by placing additional flex orders. This is elaborated in sections 2.3.3.4 and 4.3.3.1. |

##### Limit Connections

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | DSO | Operate | NA | Limit Connections | |
| **Mapping name** | DSO\_LIMIT\_CONNECTIONS |
| **Context** | When congestion issues in the operate phase cannot be solved by placing additional flex orders, the issues can be solved by limiting connections in the system.  This Pluggable Business Component limits connections for the specified congestion point and returns the amount of power gained by limiting this congestion point. |
| **Input** | * String CONGESTION\_POINT\_ENTITY\_ADDRESS   The entity address associated with the congestion point. |
| **Output** | * Long POWER\_DEFICIENCY   The amount of power in Watt gained by limiting this congestion point. |
| **References** | See [2]: chapter 5.4.8 DsoOperateCoordinator  See [1]: sections 2.3.3.4 and 4.3.3.1. |

##### Restore Connections

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | DSO | Operate | NA | Restore Connections | |
| **Mapping name** | DSO\_RESTORE\_CONNECTIONS |
| **Context** | When the load no longer exceeds the connection capacity limit the DSO can restore the connections that have been limited earlier.  This Pluggable Business Component restores the connections that were previously limited for the specified congestion point, period and PTU index. |
| **Input** | * String CONGESTION\_POINT\_ENTITY\_ADDRESS   Entity address associated with the congestion point   * org.joda.time.LocalDate PERIOD   PTU day   * Integer PTU\_INDEX   PTU Index |
| **Output** | None |
| **References** | See [2]: chapter 5.4.8 DsoOperateCoordinator  See [1]: sections 2.3.3.4 and 4.3.3.1. |

##### Initiate Settlement

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | DSO | Settlement | NA | Validate if s/D-prognoses Were Met | |
| **Mapping name** | DSO\_INITIATE\_SETTLEMENT |
| **Context** | At the start of the settlement process a DSO generates settlement items based on the flex orders of previous month and the actual power consumption on the applicable connections. For each settlement item, the delivered flex quantity and power deficiency for the PTU is calculated as well as a possible penalty.  This Pluggable Business Component prepares the settlement for the specified period based on the provided prognoses, flex requests, offers and orders and the metered data. It returns a settlement object that contains all settlement items for the supplied flex orders. |
| **Input** | * org.joda.time.LocalDate START\_DATE   Start date of the period to initiate settlement for.   * org.joda.time.LocalDate END\_DATE   End date of the period to initiate settlement for.   * List< energy.usef.core.workflow.dto.PrognosisDto> PROGNOSIS\_DTO\_LIST   The list of all relevant prognoses in the specified period.   * List< energy.usef.core.workflow.dto.FlexRequestDto> FLEX\_REQUEST\_DTO\_LIST   List of all relevant flex requests in the specified period.   * List <energy.usef.core.workflow.dto.FlexOfferDto> FLEX\_OFFER\_DTO\_LIST   List of all relevant flex offers in the specified period.   * List <energy.usef.core.workflow.dto.FlexOrderDto> FLEX\_ORDER\_DTO\_LIST   List of all relevant flex orders in the specified period.   * List <energy.usef.core.data.xml.bean.message.MeterDataSet> SMART\_METER\_DATA   The metered data for the specified period |
| **Output** | * energy.usef.core.workflow.dto.SettlementDto SETTLEMENT\_DTO The settlement object that contains a list of all settlement items for the supplied flex orders. |
| **References** | See [2]: chapter 5.5.6 DsoInitiateSettlementCoordinator  See [1]: The Settlement process between DSO and Aggregator is elaborated in sections 2.3.4.8 |

##### Calculate Penalty Amount

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | DSO | Settlement | NA | Calculate Flex Prices and Penalty Amounts | |
| **Mapping name** | DSO\_REQUEST\_PENALTY\_DATA |
| **Context** | At the start of the settlement process a DSO generates settlement items based on the previous month’s flex orders and the actual power consumption on the applicable connections. For each settlement item, the delivered flex quantity and power deficiency for the PTU is calculated as well as a possible penalty.  The input for this PBC is the settlement object and the PTU duration. The output is the input settlement object updated with penalty data. |
| **Input** | * energy.usef.core.workflow.dto.SettlementDto SETTLEMENT\_DTO The settlement object that contains a list of all settlement items for the supplied flex orders. * int PTU\_DURATION   The duration of a PTU expressed in minutes |
| **Output** | * energy.usef.core.workflow.dto.SettlementDto SETTLEMENT\_DTO   The input settlement object updated with penalty data. |
| **References** | See [2]: chapter 5.5.6 DsoInitiateSettlementCoordinator  See [1]: The Settlement process between DSO and Aggregator is elaborated in sections 2.3.4.8 |

##### Generate Connection Meter Events

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | DSO | Settlement | NA | Collect data for relevant connections | |
| **Mapping name** | DSO\_METER\_DATA\_QUERY\_EVENTS |
| **Context** | In the settlement process between the DSO and its customers, the DSO retrieves capacity reduction and outage events from the MDC, in order to compensate its customers for those events. If this data cannot be retrieved form the MDC, the DSO generates these meter events itself.  This Pluggable Business Component generates a list of capacity reduction and outage meter events for the connections and period specified in the input. |
| **Input** | * List <String> CONNECTION\_LIST   The entity addresses for connections for which meter events should be generated.   * org.joda.time.LocalDate PERIOD   PTU day |
| **Output** | * List <energy.usef.core.workflow.dto ConnectionMeterEventDto> CONNECTION\_METER\_EVENT\_DTO\_LIST   List of generated meter events for the specified connections. |
| **References** | See [2]: chapter 5.5.9 DsoCollectOrangeRegimeDataCoordinator  See [1]: Settlement between DSO and prosumers in orange regime is described in section 4.3.4. |

##### Determine Outage Durations

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | DSO | Settlement | NA | Determine outage durations | |
| **Mapping name** | DSO\_DETERMINE\_ORANGE\_OUTAGE\_PERIODS |
| **Context** | In the settlement process between the DSO and its customers, the smart meter capacity reduction and outage meter events gathered by the Meter Data Company are converted into capacity reduction and outage periods.  This Pluggable Business Component checks the data against internal records, determining if a sequence of events classifies as a capacity reduction or outage, and emits a list of actual outage periods. |
| **Input** | * List <energy.usef.core.workflow.dto.ConnectionMeterEventDto> CONNECTION\_METER\_EVENTS   List of capacity reduction and outage meter events |
| **Output** | * List <energy.usef.dso.workflow.dto.ConnectionCapacityLimitationPeriodDto> CONNECTION\_METER\_EVENT\_PERIODS   List of outage periods. |
| **References** | See [2]: chapter 5.5.10 DsoDetermineCapacityLimitationPeriodsCoordinator  See [1]: Settlement between DSO and prosumers in orange regime is described in section 4.3.4. |

##### Determine Reduction Periods

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | DSO | Settlement | NA | Determine capacity reduction periods | |
| **Mapping name** | DSO\_DETERMINE\_ORANGE\_REDUCTION\_PERIODS |
| **Context** | In the settlement process between the DSO and its customers, the smart meter capacity reduction and outage meter events gathered by the Meter Data Company are converted into capacity reduction and outage periods.  This Pluggable Business Component checks the data against internal records, determining if a sequence of events classifies as a capacity reduction or outage, and emits a list of actual reduction periods. |
| **Input** | * List <energy.usef.core.workflow.dto.ConnectionMeterEventDto> CONNECTION\_METER\_EVENTS   List of capacity reduction and outage meter events |
| **Output** | * List <energy.usef.dso.workflow.dto.ConnectionCapacityLimitationPeriodDto> CONNECTION\_METER\_EVENT\_PERIODS   List of reduction periods. |
| **References** | See [2]: chapter 5.5.10 DsoDetermineCapacityLimitationPeriodsCoordinator  See [1]: Settlement between DSO and prosumers in orange regime is described in section 4.3.4. |

##### Calculate Compensations

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | DSO | Settlement | NA | Calculate compensations | |
| **Mapping name** | DSO\_DETERMINE\_ORANGE\_REGIME\_COMPENSATIONS |
| **Context** | In the settlement process between the DSO and its customers, the capacity reduction and outage periods are used by the DSO to calculate compensations due to each prosumer.  This Pluggable Business Component calculates the compensations and posts them to an accounting system. |
| **Input** | * List <energy.usef.dso.workflow.dto.ConnectionCapacityLimitationPeriodDto> CONNECTION\_CAPACITY\_LIMITATION\_PERIOD\_DTO\_LIST   List of reduction and outage periods. |
| **Output** | none |
| **References** | See [2]: chapter 5.5.10 DsoDetermineCapacityLimitationPeriodsCoordinator  See [1]: Settlement between DSO and prosumers in orange regime is described in section 4.3.4. |

#### Meter Data Company

##### Meter Data Query

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Identification** | |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Phase** | **Regime** | **USEF step** | | MDC | Settlement | NA | Gather Requested Smart Meter data | |
| **Mapping name** | MDC\_METER\_DATA\_QUERY |
| **Context** | A participant can request the actual power values or the meter events for a set of connections. This data is used in the settlement phase to correctly calculate penalties.  This Pluggable Business Component retrieves the meter data query type from the input and returns the requested meter data for the specified connections during the specified date range (including the specified start and end dates).  For query type USAGE, the PBC populates the Power values for each connection for each PTU in the specified date range.  For query type EVENTS, the PBC populates the meter events for each connection for each day in the specified data range. |
| **Input** | * energy.usef.core.workflow.dto.MeterDataQueryTypeDto META\_DATA\_QUERY\_TYPE   The meter data query type (ANY, USAGE (default), EVENTS)   * Integer PTU\_DURATION   The duration of a PTU.   * org.joda.time.LocalDate DATE\_RANGE\_START   The start Date of the requested data. This date is included.   * org.joda.time.LocalDate DATE\_RANGE\_END   The end Date of the requested data. This date is included.   * List <String> CONNECTIONS   The list of entity addresses of connections for which data is requested. |
| **Output** | * List <energy.usef.mdc.dto.MeterDataDto> METER\_DATA\_DTO\_LIST   Output list of meter data objects per requested day. |
| **References** | See [2]: chapter 5.5.11 MdcMeterDataQueryCoordinator  See [1]: The Settlement process between DSO and Aggregator is elaborated in sections 2.3.4.8. Settlement between DSO and prosumers in orange regime is described in section 4.3.4. |

# Tools

## Database initializer

To enable injection of data in the database, a sample SQL script is provided in the USEF reference implementation delivery How to use this script is part of [3].

## Message injector

The message injector offers a REST service to send a test message, and receive an automated response indicating that the message was successfully received.

The endpoint for this is defined in energy.usef.core.endpoint.MessageEndpoint.

# Appendix - PBC implementation catalogue

## PBC Feeder

The reference implementation is delivered with a component that delivers consistent and realistic data for all PBCs in order to have USEF run with meaningful and predictable outcomes for testing and convincing demonstrations. This component is the PBC Feeder that will, on request, deliver the required input data for the separate PBC implementations that are delivered in the reference implementation. Data should be delivered for the requested PTUs.

### Input data

The data that is supplied by the PBC Feeder is based on a file that is read at system start up. The file is located in:

* usef-build\usef-deployments\usef-deployment-pbcfeeder\src\test\resources\stubinputdata.xls

This sheet only supports PTUs with a length of 15 minutes. When using the PBC Feeder the system configuration must work with the same PTU length.

The following types of data are related to or managed by the PBC Feeder:

* UncontrolledLoad

The UncontrolledLoad is defined for each of the three identified congestion points. It contains values in Watt for a week with a time step of 15 minutes. Three random factors are added/subtracted to these values to create unique load profiles for each connection:

* Each connection has a scaling factor for the Baseline that is valid for all PTUs. This factor will be between 80% and 120%.
* A random factor on top of the baseline to be generated every PTU between -10% and 10% of the scaling factor
* A random peak factor with a probability of 5% of a load random between 500W and 2000W.

No separate forecast data is provided for the uncontrolled load, therefore the actual numbers can be used as both the forecast as the realization in the operate phase.

* PVLoad and PVLoadForecast

PV (Photovoltaics) information is provided in both a forecast (PVLoadForecast) and actuals (PVLoad). The forecast should be used in the Plan and Validate phase while the actuals should be used in Operate.

Both the PVLoad and PVLoadForecast are provided on a 15 minute basis for 7 days. In general the forecast and actuals do not differ much, the forecast is more smoothly. Only for the final two days there is a significant deviation between forecast and actuals. Day 6 the forecast is 20% higher and Day 7 the forecast is 20% lower.

Each connection has the same output with a peak production of 2500W.

* FlexPotential

Every connection will be equipped with one magic flexible device that can be turned on and off at all times at the discretion of the Aggregator. This flexible device is capable of consuming or producing (50/50 distribution) between 500W and 2000W.

The magic device is initiated for every connection and is valid for all PTUs.

* Day-ahead market prices

Market prices (ApxPrices) will be delivered for every PTU. Provided price information is fictional and based on the Dutch APX market.

* Congestion Point limits

Congestion points have consumption and production limits, when these values are exceeded the Grid Safety Analysis ‘fails’. The ConsumptionLimit is configurable percentage of the maximum UncontrolledLoad that will occur at the congestion point. The ProductionLimit is the maximum net production (PVLoad-UncontrolledLoad) that will occur at the congestion point.

A suggested default value for the two configurable percentages is 90%.

How this data is used is described in the stub implementation.

### Interface

PBCFeederEndpoint:

This endpoint is used to retrieve data from the PBC Feeder:

Get all data for a specific (number of) PTU(s).

* URL: /PBCFeeder/ptu/  
  HTTP method: GET  
  Parameters: {date (date)}/{startIndex (int)}/{amount (int)}  
   date: use the following format: YYYY-MM-DD.  
   startIndex : the ptuindex of the day to start with.  
   amount : the amount of PTUs to be generated.  
  Returns: List <energy.usef.pbcfeeder.dto.PbcStubDataDto> (JSON object)

Get the uncontrolled load for a specific congestion point.

* URL: /PBCFeeder/congestionpoint/  
  HTTP method: GET  
  Parameters: {id (Integer)}  
   id: Id of the congestion point.  
  Returns: List <Double> (JSON object)

Get all APX values.

* URL: /PBCFeeder/apx  
  HTTP method: GET  
  Parameters: none  
  Returns: List <Double> (JSON object)

Get all PV actual values.

* URL: /PBCFeeder/pvactual  
  HTTP method: GET  
  Parameters: none  
  Returns: List <Double> (JSON object)

Get all PV forecast values.

* URL: /PBCFeeder/pvforecast  
  HTTP method: GET  
  Parameters: none  
  Returns: List <Double> (JSON object)

## Aggregator

### Create Elements

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.agr.workflow.step.AgrUpdateElementDataStoreStub |
| **Project** | usef-simulation-agr |
| **Description** | This stub performs the following steps to create its profile:   * Retrieve the UncontrolledLoad and PVLoadForecast for the input Period from the PBC Feeder for all input Connections. * Generate output list of elements with added profile power values:   + 1 synthetic\_data Element for:     - uncontrolled\_load   + 1 managed\_device Element with     - profile = "PV\_xxxx" (name of this type of device)     - average\_production = PVload     - potential\_flex\_production = - PVload   + 1 managed\_device Element with     - profile = "ADS\_xxxx" (name of this type of device)     - average\_consumption = ADSload     - potential\_flex\_consumption = - ADSload |

### Create Profile

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.agr.workflow.step.AgrCreateConnectionProfileStub |
| **Project** | usef-simulation-agr |
| **Description** | This stub performs the following steps to create its profile:   * Retrieve the profile power values from the supplied elements per connection map. * Generate output list of connections with added profile power values |

### Create Udi

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.agr.workflow.step.AgrCreateUdiStub |
| **Project** | usef-simulation-agr |
| **Description** | The stub implementation of the PBC creates UDI devices based on the provided elements. The “Profile” field of the elements is used to determine the type of device, for which the capabilities are defined in a JSON file.  The template for this JSON file can be found in:  usef\_ri\usef-environment\template\capabilities.json  During deployment for each aggregator this template file is copied to the configuration folder where it can be adjusted:  usef\_ri\usef-environment\nodes\localhost\configuration\<AGR\_X>\capabilities.json |

### Collect Forecast

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.agr.workflow.step.AgrCreateNDayAheadForecastStub |
| **Project** | usef-simulation-agr |
| **Description** | This stub performs the following steps to create its forecast:   * Retrieve the UncontrolledLoad, PVLoadForecast and ADSLoad for the input Period from the PBCFeeder for all input Connections. * Generate output list of connections with added forecast power values on connection and UDI level:   + forecast\_uncontrolled\_load is stored on connection level   + the PVLoad is stored in the UDI for PV:     - forecast\_average\_production = PVload     - forecast\_potential\_flex\_production = - PVload   + the ADSLoad is stored in the UDI for ADS:     - forecast\_average\_consumption = ADSload     - forecast\_potential\_flex\_consumption = - ADSload |

### Non-UDI Collect Forecast

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.agr.workflow.step.AgrNonUdiCreateNDayAheadForecastStub |
| **Project** | usef-simulation-agr |
| **Description** | This stub performs the following steps to create its forecast:   * Retrieve the UncontrolledLoad, PVLoadForecast and ADSLoad for the input Period from the PBCFeeder for all input Connections. * Generate output list of connections with added forecast power values on connection and UDI level:   + forecast\_uncontrolled\_load is stored on connection level   + the PVLoad and ADSLoad are added and stored on connection level:     - If value >= 0:  forecast\_average\_production = PVload + ADSload forecast\_potential\_flex\_production = - (PVload + ADSload)     - Else forecast\_average\_consumption = PVload + ADSload forecast\_potential\_flex\_consumption = - (PVload + ADSload) * Generate output list of connections with added forecast power values. |
|  |  |

### Optimize AGR Portfolio

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.agr.workflow.step.AgrReOptimizePortfolioStub |
| **Project** | usef-simulation-agr |
| **Description** | The stub performs the following steps to re-optimize the portfolio:   * For each connection group, calculate the difference between the latest prognoses, the current portfolio and the relevant previously unprocessed flex orders. The result is per connection group and per PTU the desired change that needs to be processed in the portfolio. * For each PTU, the summed power is divided over all relevant UDIs with potential flex and processed in the forecasted power of these UDIs.   After that, the following steps are performed to generate device messages   * Iterate through the list with desired power change per connection group and PTU and compare these with the UDI events of all UDIs for all connections on the portfolio. * For each PTU, iterate through the list of UDI events and create a device message corresponding with the UDI event under evaluation when the power of the UDI event doesn’t exceed the desired amount for that PTU. The desired amount of power for that PTU is then reduced or increased with the amount of power requested in the device message. * The iteration stops when the amount of desired power is fulfilled.   The updated portfolio and the list of device messages is returned. |

### Non-UDI Optimize AGR Portfolio

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.agr.workflow.step.AgrNonUdiReOptimizePortfolioStub |
| **Project** | usef-simulation-agr |
| **Description** | The stub performs the following steps to re-optimize the portfolio:   * For each connection group, calculate the difference between the latest prognoses, the current portfolio and the relevant previously unprocessed flex orders. The result is per connection group and per PTU the desired change that needs to be processed in the portfolio. * For each PTU, the summed power is divided over all relevant connections with potential flex and processed in the forecasted power of these connections. * The updated portfolio is returned. |

### New Prognoses Required

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.agr.workflow.step.AgrReCreatePrognosesStub |
| **Project** | usef-simulation-agr |
| **Description** | Stub implementation of the PBC which is in charge of deciding to re-create or not A-Plans and or D-Prognoses after the re-optimization of the portfolio.  The stub implementation receives the D-Prognoses, A-Plans and current portfolio and makes the decision whether the A-Plans and/or D-Prognoses must be re-created.  The PBC sums the load per connection in the portfolio per Congestion Point. If the resulting profile deviates more than 5% for any PTU, compared to the same PTU in the previous D-prognoses, a new prognoses is required.  The PBC then sums the load for all connections, if the load in any PTU deviates more than 1000W from the load for that PTU in the previous A-plan, a new A-plan is required.  In addition, to avoid an infinite loop calling the PBC, a first random decision to interrupt it is made it the beginning of the process. |

### Create Flex Offers

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.AgrFlexOfferDetermineFlexibilityStub |
| **Project** | usef-simulation-agr |
| **Description** | This stub performs the following steps to create a list of flex offers:   * Retrieve ApxPrices from PBC Feeder for all Aggregator connections. * For each of the Flex Requests received:   + Calculate the amount of available flex per PTU by subtracting the flex already offered earlier from the potential flex amount.   + Create a flex offer with either the requested flex (available flex >= requested flex) or the available flex (available flex < requested flex).   + If apxprice > 0:     - If requested flex: offered price = between 50% and 70% of ApxPrice     - If available flex: offered price = between 70% and 90% of ApxPrice   + If apxprice <= 0:     - If requested flex: offered price = between 70% and 90% of ApxPrice     - If available flex: offered price = between 50% and 70% of ApxPrice * A mandatory expiration date is provided for each flex offer. * Return the list of generated flex offers |

### Initialize Non-UDI Clusters

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.agr.workflow.step.AgrInitializeNonUdiClustersStub |
| **Project** | usef-simulation-agr |
| **Description** | Stub implementation of the PBC which is in charge of initializing the Non-UDI cluster (e.g. PowerMatcher ™[[1]](#footnote-1)).  This stub performs the following steps to inform the Non-UDI cluster:   * Retrieve the current information available in the PowerMatcher * Compare with current common reference and only send new or delete existing BRP/congestion point/connection data to the PowerMatcher.   Note: This PBC requires a PowerMatcher to be available and will try forever in case the PowerMatcher isn't responding. |

### Retrieve ADS Goals

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.agr.workflow.step.AgrNonUdiRetrieveAdsGoalRealizationStub |
| **Project** | usef-simulation-agr |
| **Description** | Stub implementation of the PBC which is in charge of retrieving observed power values from the aggregator’s external ADS solution (e.g. PowerMatcher ™).  This stub performs the following steps:   * For each connection group, retrieve the current load values from the PowerMatcher. * Store these observed power values in the portfolio on connection group level.   Note that PowerMatcher returns a combined power value for all connections and underlying devices. This includes uncontrolled load and PV load.. |

### Determine Net Demands via ADS

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.agr.workflow.step.AgrDetermineNetDemandsStub |
| **Project** | usef-simulation-agr |
| **Description** | Stub implementation of the PBC that is in charge of retrieving and updating actual and forecast power values for all connections in the portfolio.  This stub performs the following steps:   * The ”PvLoad” and “UncontrolledLoad” for the current period are retrieved from the PBCFeeder for all connections (input, validated) * To ensure the actual and forecast values are not identical each time this PBC is executed, a small random factor is added to the values for a number of PTUs. * The aggregator portfolio is updated and set the “ActualAveragePower” for the DTUs in the previous PTU based the sum of “PvLoad”, “UncontrolledLoad” and the ”UDI-Event” entries for the DTU, using appropriate signs for production (minus) and consumption (plus).  Note that the PBCFeeder values are per PTU and that PTU-to-DTU conversion of the power value is required (also for the UDI Events). * Similarly, the “ForecastAveragePower” values for all future DTUs are set * Additionally, for each connection the stub:   + picks a single random existing “INCREASE\_CAPABILITY” and modify its max power value between -5 and 5%.   + creates a single “REPORT\_CAPABILITY“ with a unique event ID if it does not exist, or delete it if it does. * Finally, the updated connection portfolio is returned. |

### Identify Change In Forecast

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.agr.workflow.step.AgrIdentifyChangeInForecastStub |
| **Project** | usef-simulation-agr |
| **Description** | Stub implementation of the PBC which is in charge of deciding whether portfolio re-optimization should take place, based on the provided connection portfolio and period.  The PBC operates in a tick tock cadence:   * When first invoked ('tick'), it will create a forecast snapshot for the specified period * When next invoked ('tock'), it will create a new snapshot for that period, compare it against the previous snapshot for that period, determine if significant changes have occurred, and trigger the next process if so * At the end of each 'tock' invocation, the previous snapshot for the specified period is always deleted, regardless of PBC outcome   To create a snapshot, the PBC selects all PTUs from the aggregator Portfolio for which:   * A forecast Power value exists (this may involve adding DTU values together) * The PTU is in Plan, Validate or Operate phase   To compare two forecast snapshots for a period, the PBC iterates over the most recent snapshot for that period.   * For each entry, it tries to find the corresponding entry in the previous snapshot. If not found, it is ignored entirely (as are entries that are present in the previous snapshot, but not in the current one). * If the new Power value differs more than 1% (clearly identified in the source code as a user-changeable constant) from the previous Power value, the PBC sets the Changed flag in the corresponding entry to true. * If all entries have been compared, the PBC sums the Power values of all Changed==true entries. If that sum is greater than 5% of all entries, the PBC output Forecast Changed flag is set to true. Otherwise, it remains false. |

### Detect Prognoses Deviations

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.agr.workflow.step.AgrDetectDeviationFromPrognosesStub |
| **Project** | usef-simulation-agr |
| **Description** | The stub converts the per-DTU supplied forecast values to a per-PTU value based on the DTUs per PTU.  The converted forecast values are compared with the corresponding per-PTU power values from the prognosis (D-prognosis or A-plan) and indicates deviations.  If the forecast power is not within the range (corresponding prognosis +/- 5%), the PTU is added to the list of PTUs to be returned. |

### Non-UDI Detect Prognoses Deviations

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.agr.workflow.step.AgrNonUdiDetectDeviationFromPrognosesStub |
| **Project** | usef-simulation-agr |
| **Description** | The supplied forecast values are compared with the corresponding per-PTU power values from the prognosis (D-prognosis or A-plan) and indicates deviations.  If the forecast power is not within the range (corresponding prognosis +/- 5%), the PTU is added to the list of PTUs to be returned. |

### Send ADS Messages

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.AgrControlActiveDemandSupplyStub |
| **Project** | usef-simulation-agr |
| **Description** | Stub implementation of the PBC that simulates the behaviour of an Aggregator controlling Active Demand and Supply.  This stub receives as input a UDI Control Message to be processed/sent.  The UDI control message will be logged and the output is empty, which implies the message was sent successfully. |

### Send ADS Goals

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.AgrNonUdiSetAdsGoalsStub |
| **Project** | usef-simulation-agr |
| **Description** | Stub implementation of the PBC which is in charge of sending ADS goals to the aggregator’s external ADS solution (e.g. PowerMatcher ™).  This stub performs the following steps:   * Transform the supplied A-plan or D-prognosis in the applicable format. * Set the goals from the A-plan to the PowerMatcher Objective Agent. * Set the goals from the D-prognosis to the PowerMatcher Congestion Manager |

### Initiate Settlement

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.agr.workflow.step.AgrInitiateSettlementStub |
| **Project** | usef-simulation-agr |
| **Description** | This stub performs the following steps to create a list of settlement items for the specified period.   * From the list of prognoses, retrieve the different connection groups. * For each connection group:   + Retrieve the flex orders and prognoses for the connection group.   + Determine the last (non-rejected) prognosis for the connection group.   + Sort the list of flex orders by creation time “descending”.   + For each PTU:     - Determine the actual power value for that PTU from the portfolio (the observed power).     - Determine the deficiency = actual power - the value of the specified PTU from the prognosis.  With this implementation, the deficiency is positive when a reduction of consumption/production is not met and negative when an increase of consumption/production is not met.     - For each flex order from the sorted list:       * Verify if the total deficiency can be caused by this flex order. If the aggregator promised to consume/produce more in the flex order, the aggregator can only have deficiency if the aggregator consumed/produced less and the other way around.  If this is the case, the absolute value of the deficiency cannot be greater than the absolute value of the flex order.       * Calculate the delivered flex order power with delivered flex order power = flex order power + deficiency.       * The remaining deficiency is processed in the remaining flex orders. In case the last flex order is processed, the remaining deficiency could be <> 0. This implies more flex was delivered then agreed upon.       * Create a settlement item for each flex order and set the correct prognosis power, actual power, delivered flex power and power deficiency. * Return the list of settlement items. |
|  |  |

### Validate Settlement Items

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.agr.workflow.step.AgrValidateSettlementItemsStub |
| **Project** | usef-simulation-agr |
| **Description** | The stub is divided in two steps.   1. Basic checks on the combination of PROGNOSIS\_SEQUENCE, PROGNOSIS\_ORIGIN and ORDER\_REFERENCE. The stub checks if a settlement has been prepared for those values in the database (DTOs object given in input). If not, the stub stops and returns a disposition DISPUTED. 2. Each prepared settlement per flex order is compared with the received FlexOrderSettlement for each PTU. Values compared are:    1. Actual Power    2. Price    3. Prognosed Power    4. Ordered Flex   A delta of 20% is accepted for these values. If this is not the case for any of them, the disposition for this FlexOrderSettlement will be set to DISPUTED. |

## Balance Responsible Party

### Received A-plan

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.brp.workflow.step.BrpReceivedAPlanStub |
| **Project** | usef-simulation-brp |
| **Description** | This stub performs the following steps to receive and process a list of A-plans:   * If an A-plan is received for the first time during the plan phase, mark the plan as processed. For each subsequent reception of the same A-plan, the chance of marking the plan as processed decreases. In that case the A-plan is accepted. * It returns a list of accepted A-plans and a list of processed A-plans for which flex trading is supposed to take place. |

### Prepare Flex Requests

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.brp.workflow.step.BrpCreateFlexRequestsStub |
| **Project** | usef-simulation-brp |
| **Description** | This stub performs the following steps to receive and process a list of A-plans:   * The ApxPrices are retrieved from the PBCFeeder. * If the A-plan is rejected, a flex request is created for each PTU in the A-plan for which the ApxPrice is among the 20% highest or lowest price range. In case of the highest price range a reduction is requested of 50% of the original amount of energy in the A-plan. In case of the lowest price range an increase is requested of 50% of theoriginal amount of energy in the A-plan. * A mandatory expiration date is provided for each flex request. * The list of flex requests is returned.     Whereas USEF messages deal with Power (values in Watts), all BRP-internal data stores should work with Energy (values in Watt-hours).   * When emitting USEF messages, all Energy values should be expressed as Power, by multiplying each value by the number of PTUs per hour. * When consuming USEF messages, all Power values should be expressed as Energy in internal data stores, by dividing each value with the number of PTUs per hour. * This stub provides a helper function that transforms a power value (Watt) of a PTU to an energy value (Watt hour). |

### Get Not Desirable Flex Orders

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.brp.workflow.step.BrpGetNotDesirableFlexOffersStub |
| **Project** | usef-simulation-brp |
| **Description** | The sequence numbers of all the flex offers which are not desirable are returned.  This stub flips a coin and returns all or none of the provided flex offers. |

### Create Flex Orders

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.brp.workflow.step.BrpPlaceFlexOrderStub |
| **Project** | usef-simulation-brp |
| **Description** | This stub performs the following steps to create a list of accepted flex offers:   * Use 2 configurable parameters accept\_threshold\_pos (default = 0.65) and accept\_threshold\_neg (default = 1.54) * When comparing prices, all flex offer prices are converted from Euro/W to Euro/MWh. * For APX prices >= 0: if the offer price is < accept\_threshold\_pos of the ApxPrice for that PTU, the offer is accepted * For APX prices < 0: If the offer price is < accept\_threshold\_neg of the ApxPrice for that PTU, the offer is accepted |

### Create Missing A-Plans

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.brp.workflow.step.BrpCreateMissingAPlansStub |
| **Project** | usef-simulation-brp |
| **Description** | This stub performs the following steps to create missing A-plans at day-ahead gate closure.  This PBC will generate a replacement A-plan with a random power value for each PTU equal to the number of AGR connections from the input times a random numbers between -500 and 500. |

### Initiate Settlement

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.brp.workflow.step.BrpInitiateSettlementStub |
| **Project** | usef-simulation-brp |
| **Description** | This stub performs the following steps to create a list of settlement items for the specified period.   * From the list of prognoses, retrieve the different connection groups. * For each connection group:   + Retrieve the flex orders and prognoses for the connection group.   + Determine the last (non-rejected) prognosis for the connection group.   + Sort the list of flex orders by creation time “descending”.   + For each PTU:     - Determine the actual power value for that PTU from the supplied meter data.     - Determine the deficiency = actual power - the value of the specified PTU from the prognosis.  With this implementation, the deficiency is positive when a reduction of consumption/production is not met and negative when an increase of consumption/production is not met.     - For each flex order from the sorted list:       * Verify if the total deficiency can be caused by this flex order. If the aggregator promised to consume/produce more in the flex order, the aggregator can only have deficiency if the aggregator consumed/produced less and the other way around.  If this is the case, the absolute value of the deficiency cannot be greater than the absolute value of the flex order.       * Calculate the delivered flex order power with delivered flex order power = flex order power + deficiency.       * The remaining deficiency is processed in the remaining flex orders. In case the last flex order is processed, the remaining deficiency could be <> 0. This implies more flex was delivered then agreed upon.       * Create a settlement item for each flex order and set the correct prognosis power, actual power, delivered flex power and power deficiency. * Return the list of settlement items. |

### Calculate Penalty Amount

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.brp.workflow.step.BrpRequestPenaltyDataStub |
| **Project** | usef-simulation-brp |
| **Description** | This stub performs the following steps to create a list of settlement objects updated with penalty data:   * The ApxPrices are retrieved from the PBCFeeder * For each Settlement object where actual power value <> prognosis value, the penalty value = APX price * The list of Settlement objects is updated with the penalty data and returned |

## Distribution System Operator

### Create Non Aggregator Forecast

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.DsoCreateNonAggregatorForecastStub |
| **Project** | usef-simulation-dso |
| **Description** | This stub performs the following steps to create a Non-Aggregator forecast:   * Based on the PTU\_DURATION the number of PTUs per day is determined. * The list of Non-Aggregator connections for the Congestion Point is retrieved from the Common Reference. * The UncontrolledLoad and PvLoadForecast are retrieved from the PBC Feeder for all connections and PTUs in the current day. * The total forecast is generated by summing up all retrieved UncontrolledLoad and PvLoadForecast for each PTU * The total forecast for all PTUs in a day for the connections not served by an Aggregator is returned. |

### Create Missing D-prognoses

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.DsoCreateMissingDPrognosisStub |
| **Project** | usef-simulation-dso |
| **Description** | This stub performs the following steps to create missing D-prognoses:   * For each PTU, a D-prognosis is created for the specified congestion point. * The power value is set equal to M\*N, where   + M is the number of AGR connections from the PBC input   + N is a random value between -500 and 500.   This list of generated D-prognoses is returned. |

### Perform Grid Safety Analysis

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.DsoCreateGridSafetyAnalysisStub |
| **Project** | usef-simulation-dso |
| **Description** | This PBC generates a Grid Safety Analysis for a congestion point based on the provided D-prognoses and Non-Aggregator forecasts.  This stub performs the following steps to generate the result:   * Retrieve ConsumptionLimit and ProductionLimit from the PBC Feeder for this congestion point. * Calculate the TotalLoad by adding up the non aggregator forecast and D-prognoses. For each PTU,  if the TotalLoad <= ConsumptionLimit OR TotalLoad >= ProductionLimit the disposition is AVAILABLE,  otherwise REQUESTED. * The associated returned power value for REQUESTED PTUs equals the TotalLoad above ConsumptionLimit OR the TotalLoad above ProductionLimit. * The associated returned power value for AVAILABLE PTUs equals the available load between TotalLoad and ConsumptionLimit OR the available load between TotalLoad and ProductionLimit. |

### Post Coloring Process

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.DsoPostColoringProcessStub |
| **Project** | usef-simulation-dso |
| **Description** | This stub takes further action, once the coloring process is done.  The current implementation has no input and output parameters and does nothing. |

### Prepare Stepwise Limiting

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.DsoPrepareStepwiseLimitingStub |
| **Project** | usef-simulation-dso |
| **Description** | In case PTUS are colored orange, a plan is created for limiting the connections in these PTUs, in order to perform stepwise connection limiting in the Operate phase.  This stub creates this plan based on the Congestion Point and the applicable PTU list.  This current implementation of the stub does nothing. |

### Create Flex Requests

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.DsoCreateFlexRequestStub |
| **Project** | usef-simulation-dso |
| **Description** | This PBC performs the following steps to generate the result:   * The DTOs from the Grid Safety Analysis already contain all the power values for REQUESTED and AVAILABLE PTUs. * This stub randomly generates 0, 1 or 2 Flex Requests for the given congestion point, day and PTUs. * A Flex Request is created if at least 2 PTUs with disposition=REQUESTED can be found. The power values from the Grid Safety Analysis input are used. If two Flex Requests are generated, the power values are divided amongst the different Requests (40/60). * A mandatory expiration date is provided for each flex request. * The list of generated Flex Requests is returned. |

### Create Flex Orders

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.DsoPlaceFlexOrdersStub |
| **Project** | usef-simulation-dso |
| **Description** | This PBC performs the following steps to process the flex offers:   * The ApxPrices are retrieved from the PBC Feeder * If the offer price is < 95% of the ApxPrice for that PTU, the offer is accepted * The sequence numbers of all the flex offers which are accepted are returned |

### Monitor Grid

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.DsoMonitorGridStub |
| **Project** | usef-simulation-dso |
| **Description** | For the specified congestion point, this stub generates the actual and maximum load values per-PTU and indicates if congestion is detected.  For this, the following values are generated and returned:   * Retrieve ConsumptionLimit and ProductionLimit from PBC Feeder for this congestion point. * Retrieve UncontrolledLoad from PBCFeeder for this congestion point. * Calculate the TotalLoad by subtracting the amount of limited power (input) from the UncontrolledLoad * For each PTU, if the TotalLoad <= ConsumptionLimit OR TotalLoad >= ProductionLimit the Boolean CONGESTION is TRUE, otherwise FALSE. * The ACTUAL\_LOAD is TotalLoad * The MAX\_LOAD is ConsumptionLimit OR ProductionLimit. |

### Place Operate Flex Orders

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.DsoPlaceFlexOrdersStub |
| **Project** | usef-simulation-dso |
| **Description** | This stub makes an ordered list of all supplied flex offers for the specified PTU\_INDEX:   * Sort the FlexOffer messages by their total Price: lowest to highest * Select the Power values for the PTUs with Disposition equal to Requested into the requirement list. * Iterate through the sorted FlexOffer list, accept the offer, add the appropriate Power to the requirement list, until all requirement values are > 0, or there are no more offers.   The list of (zero or more) (identifiers of) FlexOffer messages to be accepted is returned. |

### Limit Connections

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.DsoLimitConnectionsStub |
| **Project** | usef-simulation-dso |
| **Description** | This stub retrieves UncontrolledLoad from PBC Feeder for this congestion point and determines the maximum load.  This stub returns a power decrease for the specified congestion point. This is a random number between 50% and 75% of the maximum UncontrolledLoad. |

### Restore Connections

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.DsoRestoreConnectionsStub |
| **Project** | usef-simulation-dso |
| **Description** | When the load no longer exceeds the connection capacity limit the DSO can restore the connections that have been limited earlier.  This stub restores the connections that were previously limited for the specified congestion point, period and PTU index.  The current implementation of this stub does nothing. |

### Initiate Settlement

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.DsoInitiateSettlementStub |
| **Project** | usef-simulation-dso |
| **Description** | This stub performs the following steps to create a list of settlement items for the specified period.   * From the list of prognoses, retrieve the different connection groups. * For each connection group:   + Retrieve the flex orders and prognoses for the connection group.   + Determine the last (non-rejected) prognosis for the connection group.   + Sort the list of flex orders by creation time “descending”.   + For each PTU:     - Determine the actual power value for that PTU from the supplied meter data.     - Determine the deficiency = actual power - the value of the specified PTU from the prognosis.  With this implementation, the deficiency is positive when a reduction of consumption/production is not met and negative when an increase of consumption/production is not met.     - For each flex order from the sorted list:       * Verify if the total deficiency can be caused by this flex order. If the aggregator promised to consume/produce more in the flex order, the aggregator can only have deficiency if the aggregator consumed/produced less and the other way around. If this is the case, the absolute value of the deficiency cannot be greater than the absolute value of the flex order.       * Calculate the delivered flex order power with delivered flex order power = flex order power + deficiency.       * The remaining deficiency is processed in the remaining flex orders. In case the last flex order is processed, the remaining deficiency could be <> 0. This implies more flex was delivered then agreed upon.       * Create a settlement item for each flex order and set the correct prognosis power, actual power, delivered flex power and power deficiency. * Return the list of settlement items. |

### Calculate Penalty Amount

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.DsoRequestPenaltyDataStub |
| **Project** | usef-simulation-dso |
| **Description** | This PBC performs the following steps to process the flex offers:   * The ApxPrices are retrieved from the PBC Feeder. * The power in flex offers is in Watt for one ptu, the price in flex offers is per ptu. The APX price is in MWh. * For each Settlement object where actual power value <> prognosis value, the penalty value = (APX price / 1.000.000 / 4) \* power deficiency * The list of Settlement objects is updated with the penalty data and returned |

### Generate Connection Meter Events

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.DsoMeterDataQueryEventsStub |
| **Project** | usef-simulation-dso |
| **Description** | This stub implementation generates meter data for the specified connections during the specified date range (including the specified start and end dates) with the following values:   * Type= CAPACITY\_MANAGEMENT (33% chance) with random power value between -500 and 500 * Type= CONNECTION\_INTERRUPTION (33% chance) * Type= CONNECTION\_RESUMPTION (33% chance) |

### Determine Outage Durations

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.DsoDetermineOrangeOutagePeriodsStub |
| **Project** | usef-simulation-dso |
| **Description** | This stub implementation generates a list of outage periods based in the connection meter events supplied in the input. The stub matches outage start and end events to create the outage periods. |

### Determine Reduction Periods

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.DsoDetermineOrangeReductionPeriodsStub |
| **Project** | usef-simulation-dso |
| **Description** | This stub implementation generates a list of reduction periods based in the connection meter events supplied in the input. The stub matches reduction start and end events to create the reduction periods. |

### Calculate Compensations

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.dso.workflow.step.DsoDetermineOrangeRegimeCompensationsStub |
| **Project** | usef-simulation-dso |
| **Description** | In the settlement process between the DSO and its customers, the capacity reduction and outage periods are used by the DSO to calculate compensations due to each prosumer.  This stub calculates the compensations and posts them to an accounting system.  The current implementation of this stub does nothing. |

## Meter Data Company

### Meter Data Query

|  |  |
| --- | --- |
| **Implementation 1** | |
| **Class** | energy.usef.mdc.workflow.step.MdcMeterDataQueryStub |
| **Project** | usef-simulation-mdc |
| **Description** | This stub implementation retrieves the meter data query type from the input and returns the requested meter data for the specified connections during the specified date range (including the specified start and end dates).   * For query type USAGE, the stub retrieves the UncontrolledLoad from the PBC Feeder for each connection provided in the input. The stub generates a random delta load value between -50 and +50 for each PTU for each connection. * For query type EVENTS, the stub flips a coin and in 50% of the case it populates the meter events for each connection for each day in the specified data range with the following values:   + Type= CAPACITY\_MANAGEMENT (33% chance) with random power value between -500 and 500   + Type= CONNECTION\_INTERRUPTION (33% chance)   + Type= CONNECTION\_RESUMPTION (33% chance) |

# Appendix - PBC implementation manual

This chapter describes how to create custom PBC implementations and replace the default PBC implementations of the USEF reference implementation.

For testing purposes, new implementations of the PBCs can be added to the existing project specific for the implementer’s role (for example: usef-simulation-agr).

* For the new custom PBCs, perform the following steps:

1. Create a class implementing the WorkflowStep interface. Please use a class name that starts with <Role><Logical Name>.
2. Implement the WorkflowStep.invoke method with the desired business logic using the input and output parameters in the WorkflowContext as specified per PBC in section 4.3.2.
3. Adjust the pbc-catalog.properties file (located in the \src\main\resources folder of the project specific for the implementer’s role) and change the appropriate line for the custom PBC and specify the full class name of the custom PBC implementation. This ensures that the custom PBC implementation is used instead of the predefined USEF RI implementation.

* Re-run the cleanup and prepare scripts as documented in [3].

A more robust way of working is to use the project usef-impl.ri.usef.energy, which is available on USEF’s GitHub repository using the following link:

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

To use this project and start creating your own PBCs, first install and deploy the RI as described in [3].

* After that, new implementations of the PBCs can be implemented as described above and added to the usef-impl project specific for the implementer’s role (for example: usef-impl-agr). The usef-impl project contains links to the RI project, which are used when creating a new application.
* Instead of re-running cleanup and prepare scripts, create the application containing your new PBCs and deploy it either on the generated RI environment or on your own environment.

Example of a PBC implementation:

/\*\*

\* DsoLimitConnectionsStub.

\*/

public class DsoLimitConnectionsStub implements WorkflowStep {

private static final Logger LOGGER =

LoggerFactory.getLogger(DsoLimitConnectionsStub.class);

private static final Random RANDOM = new Random();

private static final String CONGESTION\_POINT\_ENTITY\_ADDRESS = "CONGESTION\_POINT\_ENTITY\_ADDRESS";

private static final String POWER\_DEFICIENCY = "POWER\_DEFICIENCY";

private static final int POWER\_MIN = 10000;

private static final int POWER\_MAX = 90000;

/\*\*

\* {@inheritDoc}

\*/

@Override

public WorkflowContext invoke(WorkflowContext context) {

LOGGER.debug("Starting workflow step 'Limit Connections'.");

LOGGER.debug("Parameters given in the context:\n # congestion point: {}\n",

context.getValue(CONGESTION\_POINT\_ENTITY\_ADDRESS));

context.setValue(POWER\_DEFICIENCY, new Long(POWER\_MIN +

RANDOM.nextInt(POWER\_MAX - POWER\_MIN)));

LOGGER.debug("Ending successfully workflow step 'Limit Connections'.");

return context;

}

}

Note that a PBC call is synchronous and runs within a transaction. The PBC implementation should return a result within the time boundaries of that transaction or the transaction will be rolled back, stopping the USEF process step.

If the PBC implementation interacts with other systems outside USEF, the implementer is responsible for correct and robust implementation of this interaction with regards to e.g. timing, message order, failure, parallel calls, etc.

1. PowerMatcher is a registered trademark of TNO [↑](#footnote-ref-1)