Open Reference Implementation V1.0

Deliverable D3.5

29/08/2019

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°731289

ID & Title :	Open Reference Implementation		
Version :	V1.0	Number of pages :	33
Short Description			
Revision history			
Version	**Date**	**Modifications' nature**	**Author**
V0.1	01.06.2019	Document initialized and structure created	Jonas Baude
V0.2	17.06.2019	First draft	Jonas Baude
V0.3	31.07.2019	Merging WP feedback	Jonas Baude
V0.4	08.08.2019	Final draft for TC review	Amir Ahmadifar
V0.5	27.08.2019	Finalization	Jonas Baude
V1.0	29.08.2019	Submission to the EC	Jonas Baude
Accessibility			
☒Public	☐ Consortium + EC	☐ Restricted to a specific group + EC	☐ Confidential + EC
Owner/Main responsible			
Name(s)	**Function**	**Company**	**Visa**
Marco Cupelli		RWTH	-
Author(s)/contributor(s): company name(s)			
Jonas Baude, Marco Cupelli, Amir Ahmadifar: RWTH Aachen University Olivier Genest, Pierre Mauvy: TRIALOG			
Reviewer(s): company name(s)			
Company			
Enedis, Avacon, CEZ Distribuce, E.ON, Enexis, RWTH, TRIALOG			
Approver(s): company name(s)			
Company			
Enedis, Avacon, CEZ Distribuce, E.ON, Enexis, RWTH			
Work Package ID	WP3	**Task ID**	3.1.3

Disclaimer: This report reflects only the author's view and the Agency is not responsible for any use that may be made of the information it contains.

EXECUTIVE SUMMARY

The goal of this document is to provide an overview for the reference implementation including the general code structure of the InterFlex API platform as specified in deliverable 3.4. It provides an overview of the structure and organization of the sources as well as the setup of the cloud platform. Furthermore, as a demonstration, the request-based flexibility negotiation has been executed to verify that the API behaves as expected.

During the two phases of defining the API abstract test suite (deliverable 3.6) and the reference implementation (deliverable 3.5), a certain level of indefiniteness was identified in the API specification. This lead to the identification of some findings and their respective solutions/recommendations which could be used to extend the specification and remove the unclearness.

This document is structured in four main chapters:

Chapter 1, **Introduction**, provides an overview of the InterFlex API. Afterwards, it provides a brief summary of the respective tasks for the implementation of the proposed API platform.

Chapter 2, **Reference Implementation overview**, describes the code base structure including the software requirements and following the Fireware design principles as the reference infrastructure for the implementation.

Chapter 3, **InterFlex API Implementation,** presents the main implementation including an exemplary request-based flexibility negotiation test scenario. This chapter also lists the findings, which could improve the API specification.

Chapter 4, **Outlook**, provides a summary of the deliverable and the potential future work to extend the reference implementation of the proposed API platform.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

InterFlex	Interactions between automated energy systems and flexibilities brought by energy market players
EU	European Union
TRL	Technology Readiness Level
API	Application Programming Interface
DSO	Distribution System Operator
ATS	Abstract Test Suite
HTTP	Hypertext Transfer Protocol
PEP	Policy Enforcement Point
PDP	Policy Decision Point
IDM	Identity Management
GE	Generic Enabler
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
ID	Identifier
ACK	Acknowledgment
NACK	Negative Acknowledgment
WP	Work Package

1 INTRODUCTION

The European Union (EU) Project InterFlex (Interactions between automated energy systems and flexibilities brought by energy market players) is a response to the Horizon 2020 Call for proposals, LCE-02-2016 ("Demonstration of smart grid, storage and system integration technologies with increasing share of renewable: distribution system").

This Call addresses the challenges of the distribution system operators in modernizing their systems and business models in order to be able to support the integration of distributed renewable energy sources into the energy mix. Within this context, the LCE-02-2016 Call promotes the development of technologies with a high TRL (technology readiness level) into a higher one.

InterFlex explores pathways to adapt and modernize the electric distribution system in line with the objectives of the 2020 and 2030 climate-energy packages of the European Commission. Six demonstration projects are conducted in five EU Member States (Czech Republic, France, Germany, the Netherlands and Sweden) in order to provide deep insights into the market and development potential of the orientations that were given by the call for proposals, i.e., demand-response, smart grid, storage and energy system integration.

With Enedis as the global coordinator and CEZ Distribuce as the technical director, InterFlex relies on a set of innovative use cases. Six industrial-scale demonstrators are being set up in the participating European countries. Figure 1-1 shows a map identifying the demo sites around Europe.

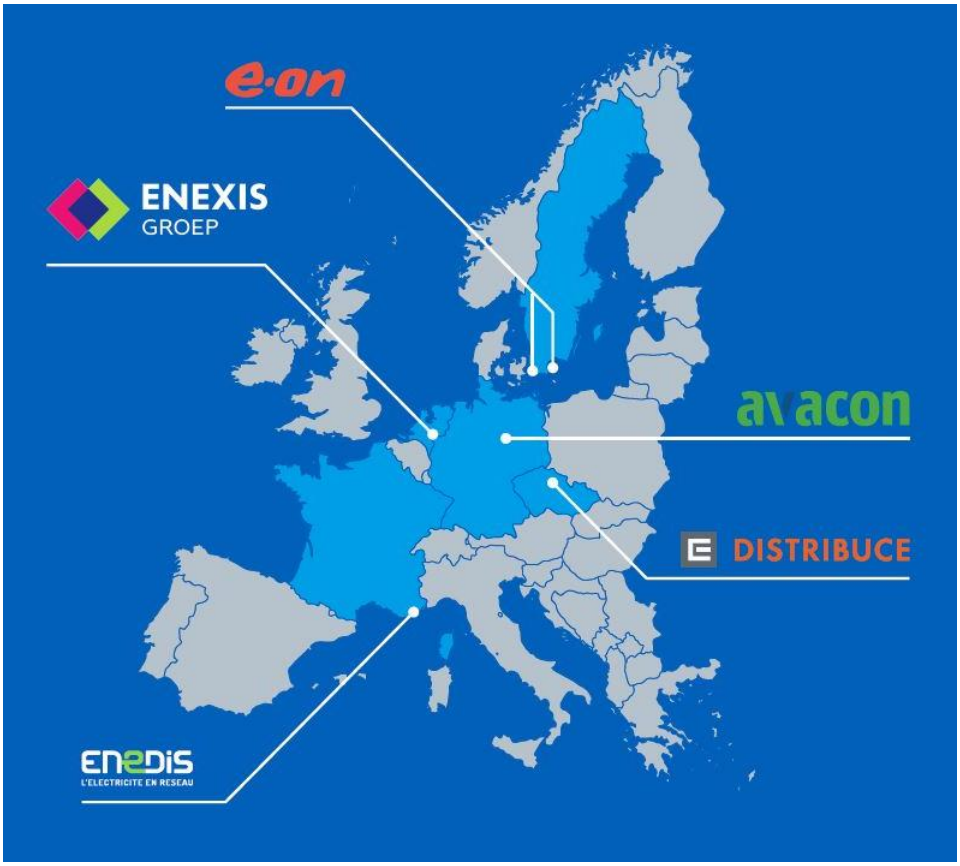

Figure 1-1 Map identifying the demo sites in the context of this project

Through these demonstration showcases, InterFlex assesses how the integration of the new solutions can lead to a local energy optimization. Technically speaking, the success of these demonstrations requires that some of the new solutions, which are today at TRLs 5-7, are further developed reaching TRLs 7-9 to be deployed in real-life conditions.

1.1 Scope of the document

This deliverable presents the reference implementation of the InterFlex API as specified in D3.4. The written report provides an overview on the structure and organization of the sources as well as the setup of the cloud platform. The implementation of the InterFlexAPI and the cloud platform are available on [1].

1.2 InterFlex API

Figure 1-2 provides a general overview on the InterFlex API and the flexibility cloud platform as described in Deliverable 3.4 [2]. The first release of the InterFlex API specification focuses on providing an interface for flexibility activation and pricing negotiation between stakeholders such as DSOs and flexibility aggregators. Following the specification in D3.4, the platform supports internal and external services. However, the reference implementation focuses on the internal key services of the platform such as flexibility activation and identity management in order to prove the feasibility of the API specified formally in Deliverable 3.4.

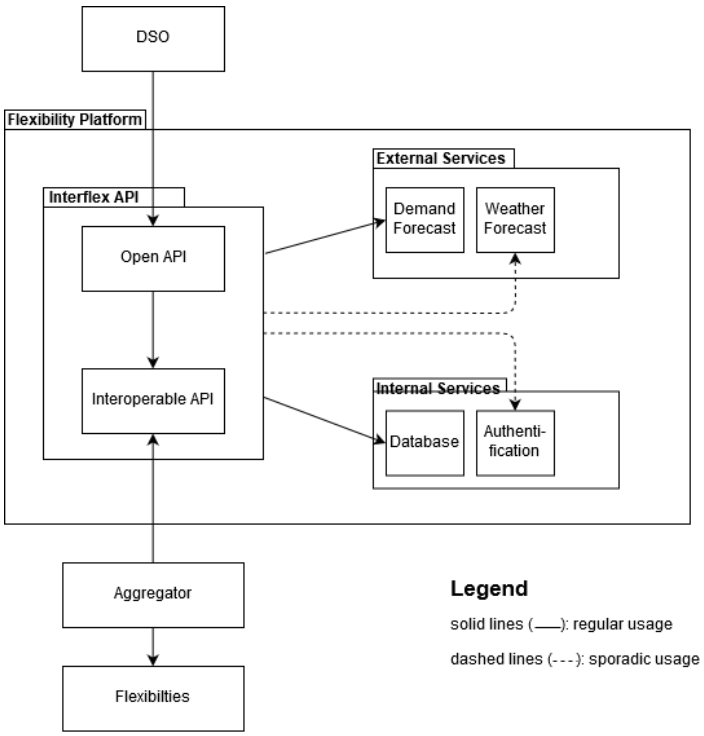

Figure 1-2 Design Overview of InterFlex API

1.3 Related Tasks

After formally specifying the API in Deliverable 3.4, the specification served as input for the Abstract Test Suite (ATS) (cf. Deliverable 3.6) definition and the reference implementation (this report). During both tasks, major and minor gray zones in the specification have been identified. These findings have been used to extend and update the specification slightly. These changes and any future updates are documented in the repository of the reference implementation.

Figure 1-3 illustrates the relation of these tasks as well as the workflow within WP3.

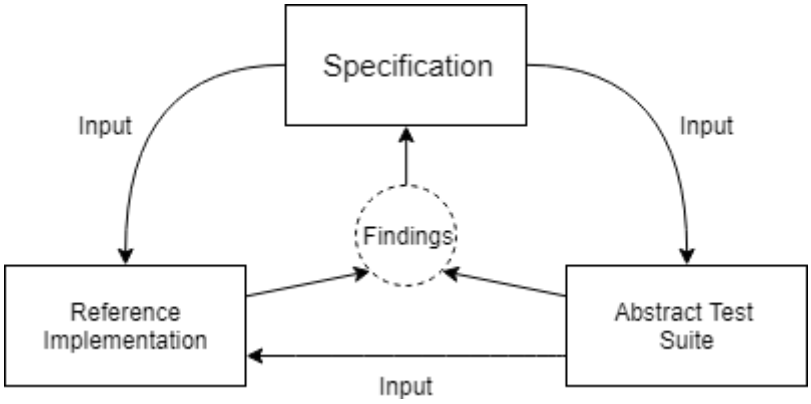

Figure 1-3 Workflow and Task Relation

1.4 Deliverable Organization

The next chapter of this report provides an overview on fundamental implementation decisions for the reference implementation of the flexibility API and on the general structure of the code base. Furthermore, it contains a brief discussion on the realization of the cloud platform and all its individual components. The third chapter presents the main implementation of the flexibility server backend and client package while focusing on the main client API functions. Furthermore, the third chapter contains a list of specification gray zones that have been identified during the ATS definition and the implementation phase. Finally, chapter four provides an outlook.

2 REFERENCE IMPLEMENTATION OVERVIEW

This chapter provides an overview on the reference implementation of the InterFlex API and the setup of the used cloud platform. Firstly, software requirements and the choice of programming languages for reference implementation are discussed. Secondly, the utilized combination of FIWARE enablers and interflex service implementations is presented. Finally, this chapter provides an overview of the code base structure for the reference implementation.

2.1 Software Requirements

The InterFlex API client is implemented as a Python 3 [3] package. Python has been chosen as it is a popular general purpose and high level programming language that is commonly used for developing desktop applications, command line tools, and web applications. It should simplify the reusability and customizability of the API. The latest version of the InterFlex API client requires a Python 3.6 interpreter (or newer) and depends on the requests [4] package for HTTP Rest APIs and the json [8] package for serializing and deserializing JSON (part of standard library).

The InterFlex API server backend is implemented in GO programming language [5]. The language has been chosen due to its native support of concurrency and due to its compilability allowing for an efficient and scalable backend implementation. Besides a variety of packages from the standard library, the backend implementation uses the gorilla/mux [6]. HTTP request router for matching API requests to their respective handler.

Both components, the InterFlex API client and the InterFlex API server backend, interact with the InterFlex flexibility platform. The reference implementation of the cloud platform is based on the open source cloud platform FIWARE [7]. The following sections contain a more detailed description of the cloud platform and the utilized FIWARE enablers.

2.2 Infrastructure

The InterFlex Flexibility Platform serves as a central instance, providing services over the interflex API. Its reference implementation is based on FIWARE [7].

Figure 2-1 depicts a set of components forming a minimal setup of the platform. Following the FIWARE design principles, the flexibility platform uses OpenStack as IaaS platform running the individual services on top.

All API clients (e.g., DSO or aggregator) connect to the Policy Enforcement Point (PEP) Proxy. It receives the client requests and queries the Identity Management (IDM) service whether the client has sufficient permissions to perform the requested action. In the current setup, the IDM also serves as a Policy Decision Point (PDP).

The PEP proxy forwards authorized client requests to the InterFlex Service, cf. Section 2.1. The InterFlex service implements the server backend of the API by receiving and forwarding client requests to the related platform services such as IDM, Database, or external services.

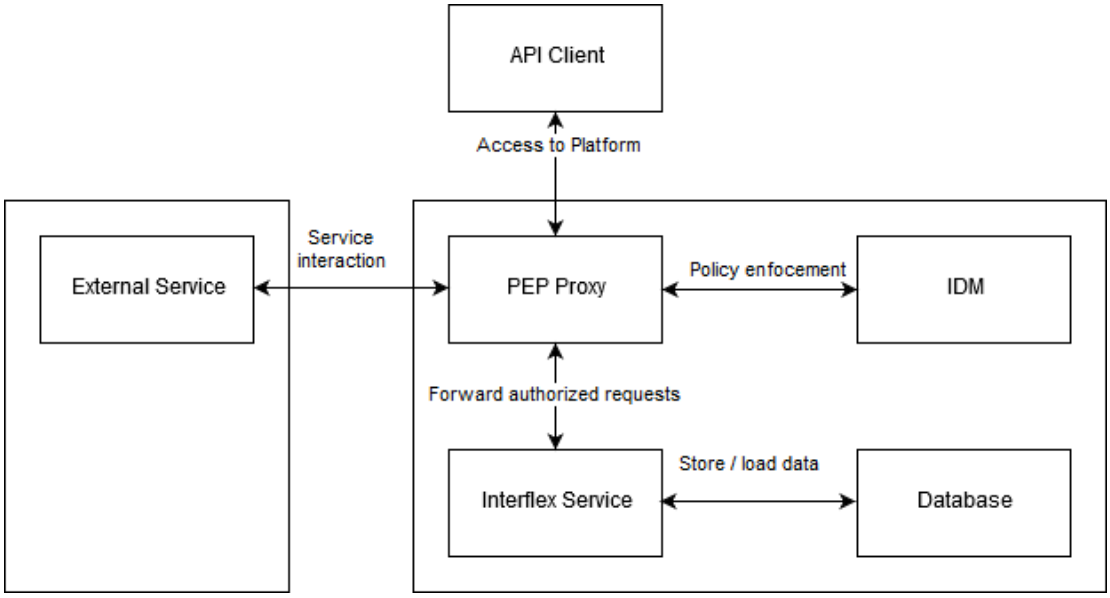

Figure 2-1 InterFlex Flexibility Platfrom Architecture

For the reference implementation, a set of so-called Generic Enablers (GEs) from the FIWARE catalogue [7] have been used to realize general cloud platform functionality such as identity management. Table 1 states the chosen generic enablers.

Service	Implementation	Language	Description
API Client	InterFlex API Client	Python	Client reference implementation
PEP Proxy	Wilma	JavaScript	Generic FIWARE Enabler
IDM	Keyrock	JavaScript	Generic FIWARE Enabler
InterFlex Service	InterFlex API Server	GO lang	Server reference implementation, considerable as domain specific FIWARE Enabler
Database	FIWARE Orion	C++	Generic FIWARE Enabler

Table 1 Mapping of Services and Implementations

2.3 Structure of the Code Base

The code base available at [1] contains all components mentioned in Section 2.1 and Section 2.2. The main repository provides a general readme, documentation, and some examples on how to use the API and the platform. Furthermore, the repository contains three submodules:

- Client API
- Server Backend
- Platform

Figure 2-2 shows the general structure of the code base repository.

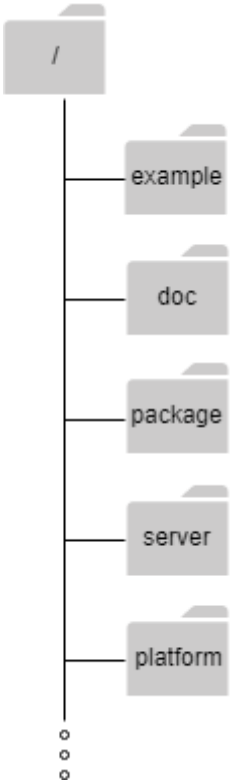

Figure 2-2 Source directory structure

2.4 Summary

The InterFlex Flexibility Platform consists of several cloud services. The presented reference implementation is based on the FIWARE cloud platform and a set of its generic enablers. The flexibility API is implemented in two components: a server and a client implementation offering the full functionality specified in Deliverable 3.4 [2]. All components are publicly available on [1].

3 INTERFLEX API IMPLEMENTATION

This chapter presents the implementation of the server backend and the client package of the InterFlex flexibility API while focusing on the interface of the client API. Furthermore, this chapter includes a list of gray zones of the formal API specification that have been identified during ATS definition phase and reference implementation phase.

3.1 Server Backend Interface Implementation

The server backend of the InterFlex API serves as a single entry point to the flexibility platform offering a RESTful HTTP API. The resources of the API are hierarchically structured in a way that allows a policy decision based the accessed URI and the applied HTTP method. The hierarchical structure for present implementation is depicted in Figure 3-1

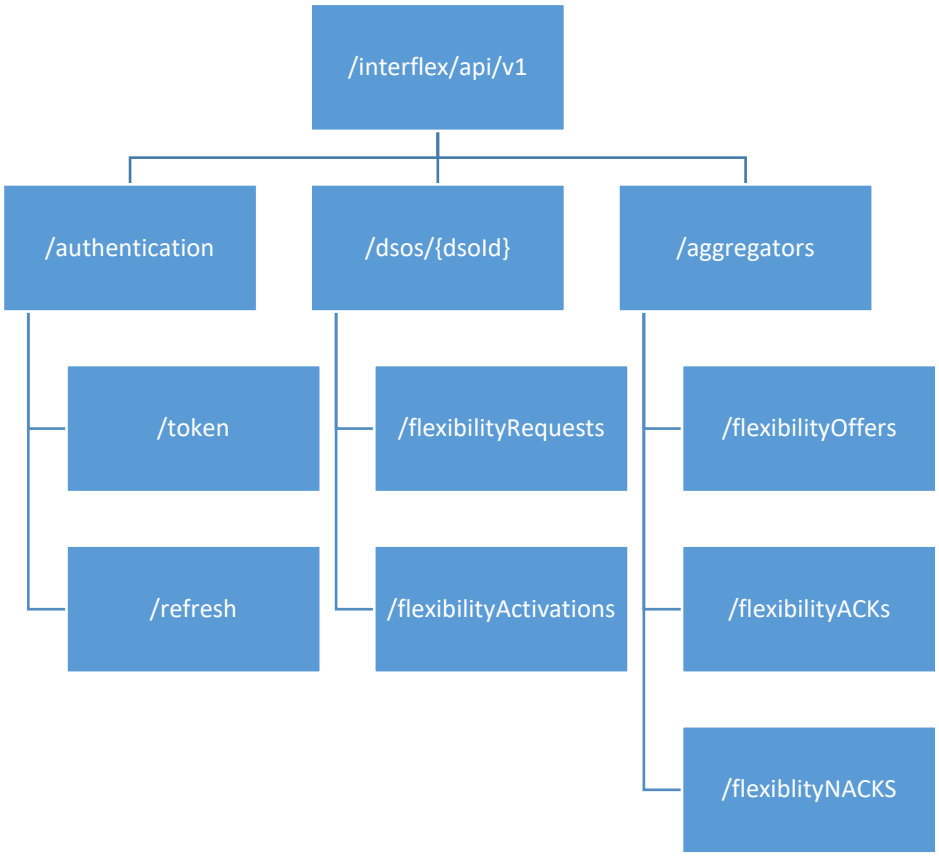

Figure 3-1 API resource structure

For example, a client can request an access token from the server backend by posting an authentication request (cf. D3.4) encoded as 'application/x-www-form-urlencoded' content as shown in Listing 1.

```
"grant_type=password&username=demo&password=demo&client_id=client&client_secret=client"
```

Listing 1 Authentication Request Body

to the URI listed in Listing 2.

```
http://hostname:port/interflex/api/v1/authentication/token
```

Listing 2 Authentication Token request URL

Upon successful authentication, the server responds a json formatted authorization response in the form shown in Listing 3.

```
{
    "access_token":"2YotnFZFEjrtGzv3JOkFSfd",
    "token_type":"bearer",
    "expires_in":3600,
    "refresh_token":"tGzv3JOkYotnFCsicMWpAA",
}
```

Listing 3 Json formatted Authentication Resonse Object

A list of all currently supported URIs is attached to the appendix of this report on page 32.

3.2 API Client Implementation

The client API reference implementation is provided as a Python package consisting of the following six modules in the latest version: authentication, flexibilityRequest, flexibilityOffer, flexibilityActivation, flexibilityActivationACK, flexibilityActivationNACK. This allows for a minimal combination of modules depending on the user needs. The following subsections provide a more detailed discussion on each module.

3.2.1 Authentication

The authentication module implements two functions allowing for authentication against the identity management of the flexibility platform, namely `authenticate()` and `refresh()`. The authenticate function expects user credentials as arguments and returns the authentication response object (cf. D3.4) containing an access token, the token type, a token expiration time, and a refresh token. The refresh function expects the latest valid refresh and returns a new authentication response. The following two tables provide the full function declarations.

3.2.1.1 `authenticate(host, port, username, user_secret, client_id, client_secret, version="v1"):`

This function sends an authentication request and returns an authentication response form the server.

Argument	Type	Description
Host	String	Hostname or IP address of the server
port	String	Port for API entry point
username	String	Username of user to be authenticated
user_secret	String	Secret of user to be authenticated
client_id	String	ID of the client application (created by IDM)
client_secret	String	Secret of the client application (created by IDM)

Table 2 Arguments of authenticate function

Return Values	Type	Description
access_token	String	Access token to be included in further requests
token_type	String	Token type, fixed to "bearer"
expires_in	Integer	Token lifetime
refresh_token	String	Refresh token that can be used to obtain a new token

Table 3 Return values of authenticate function

3.2.1.2 refresh(host, port, refresh_token, client_id, client_secret, version="v1"):

This function refreshes an access token give a valid refresh token.

Argument	Type	Description
host	String	Hostname or IP address of the server
port	String	Port for API entry point
refresh_token	String	Refresh token received with previous access token
client_id	String	ID of the client application (created by IDM)
client_secret	String	Secret of the client application (created by IDM)

Table 4 Arguments of refresh function

Return Values	Type	Description
access_token	String	Access token to be included in further requests
token_type	String	Token type, fixed to "bearer"
expires_in	Integer	Token lifetime
refresh_token	String	Refresh token that can be used to obtain a new token

Table 5 Return values of refresh function

3.2.2 Flexibility Requests

The flexibilityRequests module handles and implements flexibility requests (cf. D3.4). The module provides the following five functions:

3.2.2.1 `get_request_id(host, port, entity_id, access_token, version="v1"):`

This function fetches a unique request ID from the platform.

Argument	Type	Description
host	String	Hostname or IP address of the server
port	String	Port for API entry point
entity_id	String	Entity name of the current user, e.g. dso1
access_token	String	Valid access token
version (optional)	String	Version of API to be used (“v1” by default)

Table 6 Arguments of ret_request_id

Return Values	Type	Description
request_id	String	New and unique request ID

Table 7 Return values of get_request_id

3.2.2.2 `post_request(host, port, entity_id, access_token, request, version="v1"):`

This function posts a flexibility request to the cloud platform. The request can be created using the create_request() function. Cf. Section 3.2.2.5.

Argument	Type	Description
host	String	Hostname or IP address of the server
port	String	Port for API entry point
entity_id	String	Entity name of the current user, e.g. dso1
access_token	String	Valid access token
request	json	Json formatted flexibility request object to be posted
version (optional)	String	Version of API to be used (“v1” by default)

Table 8 Arguments of post_request

Return Values	Type	Description
server response (optional)	json	Json containing detailed error description, by default empty

Table 9 Return values of post_request

3.2.2.3 `get_all_requests(host, port, entity_id, access_token, version="v1"):`

This function returns a list of all requests from a given DSO.

Argument	Type	Description
host	String	Hostname or IP address of the server
port	String	Port for API entry point
entity_id	String	Entity name of the entity to be queried, e.g. dso1
access_token	String	Valid access token
version (optional)	String	Version of API to be used (“v1” by default)

Table 10 Arguments of get_all_requests

Return Values	Type	Description
List of all flexibilityRequests	json	Json formatted list containing all flexibility requests

Table 11 Return values of get_all_requests

3.2.2.4 `delete_request(host, port, entity_id, access_token, request_id, version="v1"):`

This function can be used to delete a request object from the cloud platform as soon as it becomes obsolete.

Argument	Type	Description
Host	String	Hostname or IP address of the server
Port	String	Port for API entry point
entity_id	String	Entity name of the current user, e.g. dso1
access_token	String	Valid access token
request_id	String	ID of the request to be deleted
version (optional)	String	Version of API to be used (“v1” by default)

Table 12 Arguments of delete_request

Return Values	Type	Description
server response (optional)	json	Json containing detailed error description, by default empty

Table 13 Return values of delete_requests

3.2.2.5 `create_request(request_id, timestamp, deadline, region_id, flexible_power, duration, unit):`

This function returns a json formatted request object in compliance with the specification in D3.4.

Argument	Type	Description
request_id	String	Valid request ID, e.g. obtained from get_request_id()
timestamp	String	Timestamp of creation (in ISO 8601 CET)
deadline	String	Deadline for offer acceptation (in ISO 8601 CET)
region_id	String	Affected grid region
flexibible_power	Double	Required amount of flexible power
duration	Double	Duration for which flexibility is required (in sec)
unit	String	Unit of flexibility

Table 14 Arguments of create_requests

Return Values	Type	Description
flexibilityRequest	json	Json formatted flexibility request object

Table 15 Return values of create_requests

3.2.3 Flexibility Offers

The flexibilityOffers module handles and implements flexibility offers (cf. D3.4). The module provides the following five functions:

3.2.3.1 `get_offer_id(host, port, entity_id, access_token, version="v1"):`

This function fetches a unique offer ID from the platform.

Argument	Type	Description
host	String	Hostname or IP address of the server
port	String	Port for API entry point
entity_id	String	Entity name of the current user, e.g. agg1
access_token	String	Valid access token
version (optional)	String	Version of API to be used (“v1” by default)

Table 16 Arguments of get_offer_id

Return Values	Type	Description
offer_id	String	New and unique offer ID

Table 17 Return values of get_offer_id

3.2.3.2 `post_offer(host, port, entity_id, access_token, offer, version="v1"):`

This function posts a flexibility offer to the cloud platform. The offer can be created using the create_offer() function, cf. Section 3.2.3.5.

Argument	Type	Description
Host	String	Hostname or IP address of the server
Port	String	Port for API entry point
entity_id	String	Entity name of the current user, e.g. dso1
access_token	String	Valid access token
Offer	json	Json formatted flexibility offer object to be posted
version (optional)	String	Version of API to be used (“v1” by default)

Table 18 Arguments of post_offer

Return Values	Type	Description
server response (optional)	json	Json containing detailed error description, by default empty

Table 19 Return values of post_offer

3.2.3.3 `get_all_offers(host, port, access_token, version="v1"):`

This function returns a list of all flexibility offers.

Argument	Type	Description
Host	String	Hostname or IP address of the server
Port	String	Port for API entry point

access_token	String	Valid access token
version (optional)	String	Version of API to be used (“v1” by default)

Table 20 Arguments of get_all_offers

Return Values	Type	Description
List of all flexibilityOffers	json	Json formatted list containing all flexibility offers

Table 21 Return values of get_all_offers

3.2.3.4 `delete_offer(host, port, entity_id, access_token, offer_id, version="v1"):`

This function can be used to delete an offer object from the cloud platform as soon as it becomes obsolete.

Argument	Type	Description
Host	String	Hostname or IP address of the server
Port	String	Port for API entry point
entity_id	String	Entity name of the current user, e.g. dso1
access_token	String	Valid access token
offer_id	String	ID of the offer to be deleted
version (optional)	String	Version of API to be used (“v1” by default)

Table 22 Arguments of delete_offer

Return Values	Type	Description
server response (optional)	json	Json containing detailed error description, by default empty

Table 23 Return values of delete_offer

3.2.3.5 `create_offer(offer_id, entity_id ,timestamp, deadline, region_id, flexible_power, granularity, duration, unit, activation_time, flex_price, flex_type, currency):`

This function returns a json formatted offer object in compliance with the specification in D3.4.

Argument	Type	Description
offer_id	String	Valid offer ID, e.g. obtained from get_offer_id()
timestamp	String	Timestamp of creation (in ISO 8601 CET)
deadline	String	Deadline for offer acceptation (in ISO 8601 CET)
region_id	String	Affected grid region
flexibible_power	Double	Required amount of flexible power
granularity	Double	Granularity the flexibility can be requested with
duration	Double	Duration for which flexibility is required (in sec)
unit	String	Unit of flexibility
activation_time	String	Time of planned activation (in ISO 8601 CET)

flex_price	Double	Flexibility Price
flex_type	String	Type of offered flexibility
currency	String	Currency

Table 24 Arguments of create_offer

Return Values	Type	Description
flexibilityOffer	json	Json formatted flexibility offer object

Table 25 Return values of create_offer

3.2.4 Flexibility Activation

The flexibilityActivation module handles and implements flexibility activations (cf. D3.4). The module provides the following five functions:

3.2.4.1 `get_activation_id(host, port, entity_id, access_token, version="v1"):`

This function fetches a unique flexibility activation ID from the platform.

Argument	Type	Description
host	String	Hostname or IP address of the server
port	String	Port for API entry point
entity_id	String	Entity name of the current user, e.g. dso11
access_token	String	Valid access token
version (optional)	String	Version of API to be used (“v1” by default)

Table 26 Arguments of get_activation_id

Return Values	Type	Description
activation_id	String	New and unique activation ID

Table 27 Return values of get_activation_id

3.2.4.2 `post_activation(host, port, entity_id, aggregator, access_token, flexibility_activation, version="v1"):`

This function posts a flexibility activation request to the cloud platform. The activation request can be created using the create_activation() function, cf. Section 3.2.4.5.

Argument	Type	Description
host	String	Hostname or IP address of the server
port	String	Port for API entry point
entity_id	String	Entity name of the current user, e.g. dso1
aggregator	String	ID of aggregator that shall receive the activation
access_token	String	Valid access token
flexibility_activation	json	Json formatted flexibility activation object to be posted
version (optional)	String	Version of API to be used (“v1” by default)

Table 28 Arguments of post_activation

Return Values	Type	Description
server response (optional)	json	Json containing detailed error description, by default empty

Table 29 Return values of post_activation

3.2.4.3 `get_all_activations(host, port, entity_id, access_token, aggregator, version="v1"):`

This function returns a list of all flexibility activation requests.

Argument	Type	Description
host	String	Hostname or IP address of the server
port	String	Port for API entry point
entity_id	String	Entity name of the current user, e.g. dso1
access_token	String	Valid access token
aggregator	String	ID of aggregator who's activations are requested
version (optional)	String	Version of API to be used ("v1" by default)

Table 30 Arguments of get_all_activations

Return Values	Type	Description
List of all flexibilityActivations	json	Json formatted list containing all flexibility activations for one particular aggregator

Table 31 Return values of get_all_activations

3.2.4.4 `delete_activation(host, port, entity_id, access_token, aggregator, activation_id, version="v1"):`

This function can be used to delete an activation object from the cloud platform as soon as it becomes obsolete.

Argument	Type	Description
host	String	Hostname or IP address of the server
port	String	Port for API entry point
entity_id	String	Entity name of the current user, e.g. dso1
access_token	String	Valid access token
aggregator	String	ID of aggregator who's activation should be deleted
activation_id	String	ID of the activation to be deleted
version (optional)	String	Version of API to be used ("v1" by default)

Table 32 Arguments of delete_activation

Return Values	Type	Description
server response (optional)	json	Json containing detailed error description, by default empty

Table 33 Return values of delete_activation

3.2.4.5 `create_activation(activation_id, timestamp, offer_id, flexible_power, granularity, duration, unit, activation_time, flex_price, flex_type, currency):`

This function returns a json formatted activation object in compliance with the specification in D3.4.

Argument	Type	Description
activation_id	String	Valid activation ID, e.g. obtained from get_activation_id()
timestamp	String	Timestamp of creation (in ISO 8601 CET)
offer_id	String	ID of related offer that was accepted
flexibible_power	Double	Required amount of flexible power
granularity	Double	Granularity the flexibility can be requested with
duration	Double	Duration for which flexibility is required (in sec)
unit	String	Unit of flexibility
activation_time	String	Time of planned activation (in ISO 8601 CET)
flex_price	Double	Flexibility Price
flex_type	String	Type of offered flexibility
currency	String	Currency

Table 34 Arguments of create_activation

Return Values	Type	Description
flexibilityActivation	json	Json formatted flexibility activation object

Table 35 Return values of create_activation

3.2.5 Flexibility Activation Acknowledgement

The flexibilityActivationACK module handles and implements flexibility activation acknowledgements (cf. D3.4). The module provides the following five functions:

3.2.5.1 `post_activation_ack(host, port, entity_id, access_token, flex_activation_ack, version="v1"):`

This function posts a flexibility activation acknowledgment to the cloud platform. The activation acknowledgment can be created using the create_activation_ack() function, cf. Section 3.2.4.5.

Argument	Type	Description
host	String	Hostname or IP address of the server
port	String	Port for API entry point
entity_id	String	Entity name of the current user, e.g. agg1
access_token	String	Valid access token
flex_activation_ack	json	Json formatted flexibility activation ACK object to be posted
version (optional)	String	Version of API to be used (“v1” by default)

Table 36 Arguments of post_activation_ack

Return Values	Type	Description
server response (optional)	json	Json containing detailed error description, by default empty

Table 37 Return values of post_activation_ack

3.2.5.2 `get_all_activation_acks(host, port, access_token, aggregator, version="v1"):`

This function returns a list of all flexibility activation acknowledgments.

Argument	Type	Description
host	String	Hostname or IP address of the server
port	String	Port for API entry point
access_token	String	Valid access token
aggregator	String	ID of aggregator who's activation ACKs are requested
version (optional)	String	Version of API to be used ("v1" by default)

Table 38 Arguments of get_all_activation_acks

Return Values	Type	Description
List of all flexibilityActivations	json	Json formatted list containing all flexibility activation ACKS for one particular aggregator

Table 39 Return values of get_all_activation_acks

3.2.5.3 `delete_activation_ack(host, port, access_token, aggregator, activation_ack_id, version="v1"):`

This function can be used to delete an activation acknowledgment object from the cloud platform as soon as it becomes obsolete.

Argument	Type	Description
host	String	Hostname or IP address of the server
port	String	Port for API entry point
access_token	String	Valid access token
aggregator	String	ID of aggregator who's activation ACK should be deleted
activation_ack_id	String	ID of the activation ACK to be deleted
version (optional)	String	Version of API to be used ("v1" by default)

Table 40 Arguments of delete_activation_ack

Return Values	Type	Description
server response (optional)	json	Json containing detailed error description, by default empty

Table 41 Return values of delete_activation_ack

3.2.5.4 `create_activation_ack(ack_id, timestamp, offer_id, flexible_power, granularity, duration, unit, activation_time):`

This function returns a json formatted activation acknowledgment object in compliance with the specification in D3.4.

Argument	Type	Description
ack_id	String	Valid activation ACK ID
timestamp	String	Timestamp of creation (in ISO 8601 CET)
offer_id	String	ID of related offer that was activated
flexibible_power	Double	activated amount of flexible power
granularity	Double	Granularity the flexibility can be requested with
duration	Double	Duration for which flexibility is required (in sec)
unit	String	Unit of flexibility
activation_time	String	Time of planned activation (in ISO 8601 CET)

Table 42 Arguments of create_activation_ack

Return Values	Type	Description
flexibilityActivationACK	json	Json formatted flexibility activation ACK object

Table 43 Return values of create_activation_ack

3.2.6 Flexibility Activation Unacknowledgement

The flexibilityActivationNACK module handles and implements flexibility activation unacknowledgements (cf. D3.4). The module provides the following five functions:

3.2.6.1 `post_activation_nack(host, port, entity_id, access_token, flexibility_activation, version="v1"):`

This function posts a flexibility activation unacknowledgment to the cloud platform. The activation unacknowledgment can be created using the create_activation_nack() function, cf. Section 3.2.4.5.

Argument	Type	Description
host	String	Hostname or IP address of the server
port	String	Port for API entry point
entity_id	String	Entity name of the current user, e.g. agg1
access_token	String	Valid access token
flexibility_activation	json	Json formatted flexibility activation object to be posted
version (optional)	String	Version of API to be used (“v1” by default)

Table 44 Arguments of post_activation_nack

Return Values	Type	Description
server response (optional)	json	Json containing detailed error description, by default empty

Table 45 Return values of psot_activation_nack

3.2.6.2 `get_all_activation_nacks(host, port, access_token, aggregator, version="v1"):`

This function returns a list of all flexibility activation acknowledgments.

Argument	Type	Description
host	String	Hostname or IP address of the server
port	String	Port for API entry point
access_token	String	Valid access token
aggregator	String	ID of aggregator who's activation NACKs are requested
version (optional)	String	Version of API to be used ("v1" by default)

Table 46 Arguments of get_all_activation_nacks

Return Values	Type	Description
List of all flexibilityActivations	json	Json formatted list containing all flexibility activation ACKS for one particular aggregator

Table 47 Return values of get_all_activation_nacks

3.2.6.3 `delete_activation_nack(host, port, access_token, aggregator, activation_ack_id, version="v1"):`

This function can be used to delete an activation unacknowledgment object from the cloud platform as soon as it becomes obsolete.

Argument	Type	Description
Host	String	Hostname or IP address of the server
Port	String	Port for API entry point
access_token	String	Valid access token
Aggregator	String	ID of aggregator who's activation NACK should be deleted
activation_ack_id	String	ID of the activation ACK to be deleted
version (optional)	String	Version of API to be used ("v1" by default)

Table 48 Arguments of delete_activation_nack

Return Values	Type	Description
server response (optional)	json	Json containing detailed error description, by default empty

Table 49 Return values of delete_activation_nack

3.2.6.4 `create_activation_nack(nack_id, timestamp, offer_id, flexible_power, granularity, duration, unit, activation_time):`

This function returns a json formatted activation acknowledgment object in compliance with the specification in D3.4.

Argument	Type	Description
nack_id	String	Valid activation NACK ID
timestamp	String	Timestamp of creation (in ISO 8601 CET)

offer_id	String	ID of related offer that was activated
flexibible_power	Double	activated amount of flexible power
granularity	Double	Granularity the flexibility can be requested with
duration	Double	Duration for which flexibility is required (in sec)
unit	String	Unit of flexibility
activation_time	String	Time of planned activation (in ISO 8601 CET)

Table 50 Arguments of create_activation_nack

Return Values	Type	Description
flexibilityActivationNACK	json	Json formatted flexibility activation ACK object

Table 51 Return values of create_activation_nack

3.3 Extensions to API Specification

While defining the abstract test suite (cf. Deliverable 3.6) and implementing the reference implementation (this document) some gray zones in the formal API specification were identified, as stated in Section 1.3. The following table provides an overview on the findings and corresponding extensions to the API specification that solve these issues.

Finding No.	Description	Identified Solutions	Chosen solution
001	It is undefined who creates IDs such as offer_id or request_id.	Solution 1: Every creating entity takes responsibility for creating a unique identifier. The structure of IDs could be specified, e.g. by specifying a fixed prefix for each entity. Solution 2: The platform provides unique identifiers upon request.	Both solutions are feasible. Solution 2 has been added to the reference implementation.
002	How should an aggregator respond to a flexibility request when it cannot offer any flexibility?	Solution 1: Aggregators that cannot provide any flexibility do not respond to the request. A DSO neglects all aggregators that do not respond within the given deadline. Solution 2: Aggregators create and post an offer with a predefined amount of flexibility (e.g. 0 or -1) to indicate that they cannot provide feasible flexibility.	Solution 2

		Solution 3: A new object could be defined to announce the renunciation for the current request.	
003	There are no rules for plausibility verification on the platform.	Solution 1: The specification could be extended with a set of plausibility checks that can be performed by the server backend.	Solution 1
004	'How can the platform prevent irrelevant offers such as already invalid ones?	Solution 1: The server-side handlers for the getter functions could be extended with a date filter and other plausibility checks (cf. finding 003)	Solution 1
005	Aggregators are not informed that they their offer was rejected by the DSO.	Solution 1: A negative acknowledgment for the aggregator could be introduced. Solution 2: A timeout for receiving an activation request could be specified.	Solution 2
006	Following the specification, only immediate flexibility activation be requested.	Solution 1: The flexibility object (cf. D3.4) that is part of the flexiblity requests, offers, and activations can be extended with an "activation_time" attribute.	Solution 1
007	Type of "duration" is not coherent between flexibilityOffers (*string*) and flexibilityRequests (*double*)	Solution 1: As the duration refers to a time span and not to a date, it should be of type *double* in both cases.	Solution 1
008	price_t defined in the Flexbility Activation Request is defined two times while the currency is missing	Solution 1: This seems to be a typo and will be updated.	Solution 1

Table 52 Listed extensions to API specification

3.4 InterFlex API Verification

In order to verify that the reference implementation of the InterFlex flexibility API is working as expected the request-based flexibility negotiation scenario that has been defined in Deliverable 3.4 was implemented using the client API. The demonstration is provided in the examples directory of the repository. The sequence chart provided in Figure 3-2 illustrates

the data flow in the API demonstrator. In order to verify all features of the API, a second aggregator was added to the verification scenario to show that the selection of one offer (in this case basically the cheapest offer) succeeds.

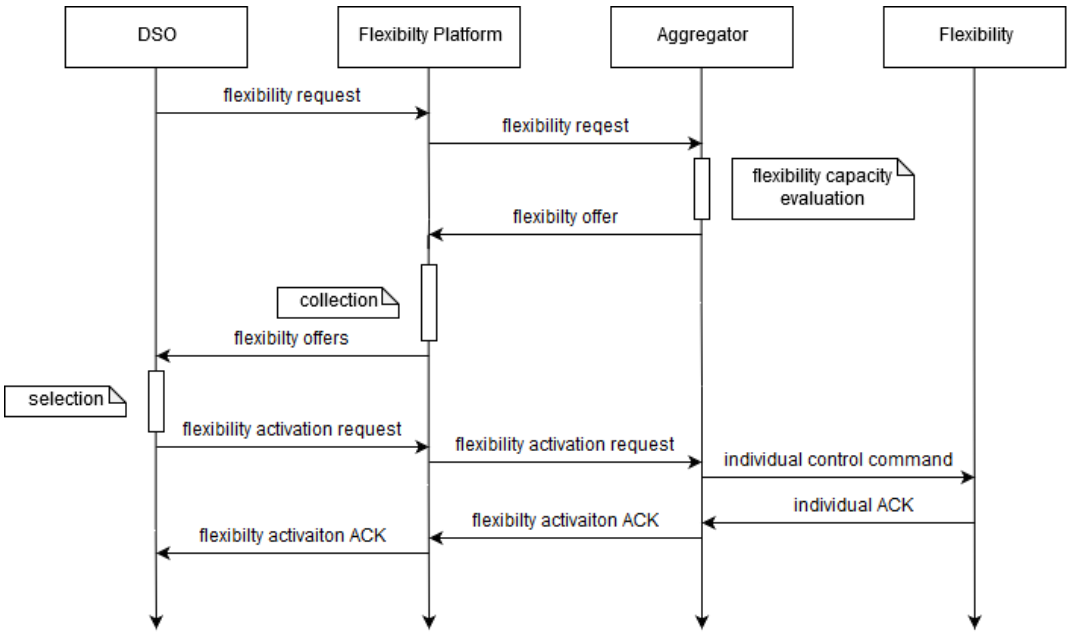

Figure 3-2 Request-based Flexibility Negotiation

3.5 Summary

This chapter presented the implementation of the server backend and the client package of the InterFlex flexibility API while focusing on the interface of the client API: its functions, the arguments, and the return values thereof. During the two phases of defining the abstract test suite and implementing the reference implementation, some major and minor ambiguities or gray zones were also identified. Section 3.3 provides a list of these findings as well as solutions to extend the specification. Finally, the request-based flexibility negotiation has been implemented as a demonstrator to verify that the API behaves as expected.

4 OUTLOOK

This document presented the reference implementation of the InterFlex API for flexibility aggregation as specified in Deliverable 3.4. In collaboration with partners within WP3, the specification was updated in order to solve identified issues or gray zones in the specification. The reference implementation has been verified with the abstract test suite (cf. Deliverable 3.6) and the verification script that implements the request-based flexibility negotiation.

The entire reference implementation, including the client interface, the server backend, and an exemplary cloud platform setup, will be publicly available on [1]. Future extensions and update to the specification will be published and documented in this repository.

Future releases could contain support for offering external services over the flexibility cloud platform, support for subscriptions in order to reduce the communication overhead with the platform, and minor bug fixes.

5 BIBLIOGRAPHY

[1] [Online] https://git.rwth-aachen.de/acs/public/deliverables/interflex/flexibilityplatfrom
[2] “InterFLEX D3.4 – Interoperable APIs Specification”, April 2019
[3] [Online] https://www.python.org/
[4] [Online] https://2.python-requests.org/en/master/#
[5] [Online] https://golang.org/
[6] [Online] https://github.com/gorilla/mux
[7] [Online] https://www.fiware.org/developers/
[8] [Online] https://docs.python.org/3/library/json.html

A. APPENDIX

Resource URI	Allowed Methods	Content	Description
/authentication/token	POST	Login credentials	Post credentials to receive authentication response
/authentication/refresh	POST	Refresh token	Post refresh token to re-authenticate
/dsos/{dso}/flexibilityRequests	POST	flexibilityRequest	Post a flexibility request to the platform
	GET	--	Receive a list of flexibility requests
/dsos/{dso}/flexibilityRequests/requestId	GET	--	Get a unique request ID from the platform
/dsos/{dso}/flexibilityRequests/{requestId}	DELETE	--	Delete the request identified by {requestId}
/dsos/{dso}/flexibilityActivations/activationId	GET	--	Get a unique activatonId from the platform
/dsos/{dso}/flexibilityActivations/{agg}	GET	--	Get all flexiblityActiavation from a particular DSO {dso} to a particular aggregator {agg}
	POST	flexibilityActivation	Post a flexibilityActivation from one DSO {dso} to a particular aggregator {agg}
/dsos/{dso}/flexibilityActivations/{agg}/{activationId}	DELETE	--	Delete a certain flexibilityActivation object referenced by {activationId}
/aggregators/flexibilityOffers	GET	--	Get a list of all flexibility offers
/aggregators/{agg}/flexibilityOffers	POST	flexibilityOffer	Post a flexibility offer
/aggregators/{agg}/flexibilityOffers/{offerId}	DELTE	--	Delete a certain flexibility offer identified by {offerId}

/aggregators/{agg}/flexibilityOffers/offerId	*GET*	--	*Get a unique offer ID*
/aggregators/{agg}/flexibilityActivations/acks	GET	--	Get a list of all activation acknowledgments
	POST	Flexibility Activation ACK	Post an activation acknowledgment
/aggregators/{agg}/flexibilityActivations/acks/{ackId}	DELETE	--	Delete a certain activation acknowledgment referred to by {ackId}
/aggregators/{agg}/flexibilityActivations/acks/ackId	GET	--	Get a unique flexibility activation acknowledgement ID from the platform
/aggregators/{agg}/flexibilityActivations/nacks	GET	--	Get a list of all activation un acknowledgments
	POST	Flexibility Activation NACK	Post a new activation unacknowledgment to the platform
/aggregators/{agg}/flexibilityActivations/nacks/{nackId}	DELETE	--	Delete a certain activation unacknowledgment referred to by {ackId}
/aggregators/{agg}/flexibilityActivations/nacks/nackId	GET	--	Get a unique flexibility activation acknowledgement ID from the platform