IT Documentation

Network Operator & Infrastructure Owner

Date: February 29, 2024

Content

1. Introduction

1.1 Purpose

This document has the purpose of establishing a common understanding of how a Network Operator will interface with the OpenNet systems to perform provisioning, capacity check and afterwards confirm and reserve capacity in the network. Addtionally the Network Operator needs to handle Work Force Management and in-life service changes and service assurance processes. The aim is to provide the Network Operator with sufficient context and detail with regard to communication with OpenNets interfaces.

It is envisaged that this high-level definition provides a sufficient architectural view with regard to the solution, approach and capabilities supported. The information herein will enable the Network Operator with an understanding as to how to interaction with OpenNet, and will occur and enable the necessary technical planning to develop systems/interfaces to consume OpenNet APIs for the operation of key business processes.

The full and final description will be developed and controlled by OpenNet, as it is the owner of the OpenNet interface.

1.2 Limitations

This document is to be considered as an API specification. For request examples see the OpenNet Connectivity packages.

1.3 Revision History

Version	**Date**	**Description**
0.9	12.02.2018	First draft
0.91	09.03.2018	Updated the document
0.92	26.04.2018	Updated
0.93	29.05.2018	Added information mostly around fulfillment, events, notes.
0.94	04.06.2018	Added information about trouble ticket API
0.95	07.06.2018	Changed text from rescheduleAfter to rescheduleFrom in event and note definitions
0.96	09.06.2018	Added SUSPEND_SLA and RESUME_SLA to the defined trouble ticket tags.
0.97	18.06.2018	As agreed the codes for Fulfillment Request type has been modified to capital letters and underscores. Clarified information where needed, and added a little more information about the internal structure of the OpenNet system.
0.98	02.07.2018	Fixed wrong name and bad spelling "requestSeqenceNo" to "requestTypeSequenceNo"
0.99	04.07.2018	In Confirm Capacity request description, reference to networkOperatorDataField1-6 has been corrected to networkOperatorOut1-6.
1.00	30.07.2018	Added missing] in the Get Trouble ticket JSON structure regarding notes array.
1.01	06.08.2018	Add note was wrongly stated to be a get. It should be a POST operation. With regard to events ORDFUL code should be ODRFUL instead.
1.12	16.10.2018	The document has a new format
1.13	19.10.2018	Added description of timeslot handling and management.
1.14	19.10.2018	Corrected trouble ticket API type field. Was "STANDARD" should be "STAN".
1.15	29.10.2018	In capacityCheckAndReserve and confirmCapacity API requests for H2 the content of the UDF field "serviceProviderIn1" has been modified to match formatting used else where. For H2 the expected format of serviceProviderIn1 is now: "{"svlan":34, "poiID":"JDMSK78986546"}"
1.16	03.12.2018	Corrected spelling mistakes. "REOPENNED" has been corrected to "REOPENED".
1.17	21.03.2019	Rewritten the Trouble Ticket chapter, to align with new features.

1.18	12.04.2019	Replaced old figures and sequence diagrams with new optimized and updated visualizations. Also corrected minor spelling mistakes.
1.19	13.10.2019	Updated H2 related info related to serviceProviderIn and serviceProviderOut UDF fields. Updated wording in "Handling H1 & H3 Sold Together" chapter, to match agreed and implemented functionality. Main change is appointmentDate or actionDate is visible on fulfillment requests for both H1 and H3 service.
1.20	17.12.2019	Improved documentation of "Support Tools". See section 11.1
1.21	03.01.2020	Added documentation for GENUPD eventSubType Renamed document to "IT Documentation - Network Operator & Infrastructure Owner" Added Outage documentation
1.22	03.01.2020	Added endpoint information for Production, and also sFTP hostname+port for both Pre-Prod and Production. Modified wording in section 8. In section 9.1.1 Capacity Check & Reservation, the mapping between inputted C-VLAN's and returned T-VLAN's has been clarified. Removed section 15, 16 and 17 covering ITSM and roles and responsibility. The information was obsolete and is now covered in process documentation and/or joint test planer. Fixed minor wording errors through out the document.
1.23	16.01.2020	Added documentation for fulfillment API version 2. Only change is adding "contactEmail" property to WORK_ORDER_NO, WORK_ORDER_IO and WORK_ORDER_PROVISIONING fulfillment requests. Added documentation for contactEmail to trouble ticket API version 2.0. URI change for capacityReservationTimedOut API. Moved from version 1.0 to 2.0 URI change for addNote API. Moved from version 1.4 to 2.0.
1.24	17.02.2020	Added IO Charging file specification. (Section 14) Updated possible timeslot codes to include AM_PR_1, AM_PR_2, PM_PR_1, PM_PR_2, PM_PR_3 Updated description of adminState in Support Tools API.
1.25	27.02.2020	Renamed section 10 to "Order fulfilment" instead of "Fulfilment requests" and reorganized sub sections. Introduced description of new event feature, where events can flow from NO->IO or IO->NO using two new event types "NOEVENT" and "IOEVENT". This also means that the getNextEvent and updateEvent API's are now introduced for NO/IO entities.
1.26	18.06.2020	Added section 19 Entity Naming Table. Specified that "null" can be returned in rx/tx value when calling support tools.
1.27	22.08-2020	Added section 18 – API Rate limits and quota policies
1.28	25.02.2021	Documentation updated to reflect default token expiery time being 5 minuttes instead of 30 minuttes.

		In the sample events, " has been changed to \" to reflect the needed escape charactors.
1.29	25.02.2021	Specified that IO's are not allowed to remove a temporary reservation unless a success response is returned from OpenNet in section 9.1.5 Added http response 4xx and 5xx to data definition in 9.1.5 Added Main Product relevance column to 11.1.2 If no qosProfile is configured in returnValue.configuration.qosProfile, then it must be null or "", updated in 11.1.2
1.30	26.02.2021	Updated Trouble Ticket chapter of this document to describe in more details the different parameters and also a new **searchOpenTroubleTickets API**. Added description of **internal trouble ticket notes** to get and update trouble ticket API description.
1.31	04.03.2021	Updated 11.1.2 – if no CVLANs are provisioned then an empty array [] or null is returned in returnValue.configuration.provisionedVlans[index].id
1.32	26.03.2021	Added mising appointment information in getTroubleTicket API for SAWA type trouble tickets.
1.33	06.04.2021	Corrected confirmCapacityReservation description in regard to H2 and expected value in serviceProviderIn1. Earlier the description described a requirement to pass S-VLAN, which was incorrect.
1.34	26.04.2021	Corrected "*Table 1: Data items – Search for Open Trouble Tickets":* "Array of "OPEN" Trouble Ticket Id's which are related to the user". It was previously "Array of Trouble Ticket Id's assigned to the user".
1.35	04.05.2021	Added entities to entity naming table
1.36	11.05.2021	Added documentation for new errorAfterInstallation property for GET Trouble Ticket API version 2.0 onwards. See section 11.2.1 for details
1.37	27-07-2021	Removed variants for H2 (BASIC, ENTERPRISE, ADVANCED) Updated requirements in regard to serviceProviderOut1 and serviceProviderOut3 properties in Capacity Check And Reserve API response in regard to H2. ***Note**: **As a minimum the serviceProviderOut1 field must be populated** for H1, **H2** and H3 services in the capacityCheckAndReserve response. **For H2 also the transceiverSpeed in serviceProviderOut3 is expected to be returned as part of the capacityCheckAndReserve Response.*** Added description of new autoAcknowledge=false feature for Get Fulfillment Request API and related changes to Update Fulfillment Request API.
1.38	Not released	Updated endpoint URI for Add Note API to version 3.0
1.39	31-08-2021	New version 3.0 of fulfillment request API described. This version change the order fulfillment requests are returned to a FIFO style queue, and also introduce a priority property in the fulfillment request collect response, which allow NO/IO to detect if the appointment/actionDate has related to the order, was scheduled using the priority=triue flag from SP side. Priority=true means that the order has been agreed between SP and IO to be fast tracked.
1.40	10.11.2021	Added information about relatedEntities record, now being available in the GET Trouble Ticket response. (Require use of API version 3.0)
1.41		Corrected Support Tools API response definitions, in regard to which values are relevant for which main products.
1.42	01.03.2022	Added scenario causing a No Capacity Available response from capacityCheck and capacityCheckAndReserve API's. **Scenario**: If an ongoing order, which involves the initial installation of a fiberbox, is currently preventing placement of new orders until completion. Expected completion date is only an indication, as the order can be rescheduled or cancelled.

		”message”:”{\”reasonCode\”:3000,\”description\”:\”Ongoing installation order on address. Expected completion 2022-05-23\”}”
1.43	03.10.2022	Clarification added to “9.1.3 Confirm Capacity Reservation”, to explicitly describe the required functionality of the Confirm Capacity Reservation API. Please take note of this, as the functionality of the API is expected to be very limited, to avoid the need for manual error handling on orders. Section 14.3: Corrected wrong format definition for transaction effective date, in regard to IO Charging file records. From yyyy-MM-dd to yyyy_MM_dd. Added note under BARRING fulfillment request definition, specifying the expected behaviour if SP executes a change of options on a barred service. Clarified that installationID in Capacity Check API as well as requiredInstallationID in serviceProviderIn UDF for Capacity Check and Reserve API is only expected to hold values which has been made available to OpenNet via the Address file. For Capacity Check and Reserve, the NO can optionally accept values in requiredInstallationID which has been registered on the address by not yet uploaded to OpenNet. NO is NOT required to build this logic. Removed option to reopen a cancelled trouble ticket. Added documentation for getServiceStatus_v2
1.44	28.11.2022	Added documentation for Trouble Ticket type “NETW”=Network ticket to support OpenNet Process 6.6 Multi and POI error. See GET Trouble Ticket API response description. Reference to Swagger documentation removed. OpenNet instead point readers to the OpenNet Connectivity Packages for SP or NO/IO entities to see request examples.
1.45	30.11.2023	Corrected SAWA trouble-ticket subState “REOPENED_SCHEDULED” to “REOPN_SCHD”. REOPENED_SCHEDULED was never implemented due to character restrictions.
1.46	28.02.2023	Added description of ratelimit headers and http 429 – Too Many Requests response within section 17. Removed requirement for VOCES or FOCES client certificates.
1.47	24.04.2023	Added description of IO’s being able to call Support Tools API which was only available to SP’s earlier. This is to improve the support of IO and NO being two different companies.
1.48	14.08.2023	Added status query parameter to Trouble Ticket Search API, to allow API to return OPEN, CLOSED or ALL updated trouble tickets since a specific time.
1.49	01.09.2023	Fixed typo in documentation for Open and Close Timeslot API’s. The documentation mistakenly wrote workFieldsArea where it should say workFieldArea. Using workFieldsArea as property name would result in the property being ignored in the timeslot filtration.
1.50		Added NETW as possible enum value for the type query parameter in Trouble Ticket Search API. Fixed technologyCode on workorders
1.51	14.02.2024	Defines Support tools response in regard to ethernet port interface for speeds higher than 1 Gbps Moved the definition of reasonCode and description in Capacity Check and Capacity Check and Reserve to the OpenNet Appendix.

1.4 Audience

People with insight in the IT solution and integration.

1.5 References

Title	Document Ref.	Issue	Dated

1.6 Abbreviations & Concept

Abbreviation	**Explanations**
API	Application Programming Interface
CRM	Customer Relation Management which relate to OpenNet Core backend system
Infrastructure Owner	Own and develop the physical network and sells access to the infrastructure via the OpenNet Entity
Network Operator	To deliver technical operations and incident handling, including 2nd level and 3rd level support as defined in potential outsourcing contract with OpenNet.
Order	An instruction received by the OpenNet Integration Layer from a Service Provider to provide, change or terminate one or more OpenNet packages.
OpenNet	Name of Wholesale OpenNet is going to deliver wholesale products to Service Providers. Has primary contact with Wholesale customers (Service Providers).
OpenNet Integration Layer	The system built and administered by OpenNet which surrounds and interacts with the Cerillion Core System, via APIs.
Service Provider	To deliver Services to end-customers on the network, providing full packages including any content, voice and other Value Added Services they may find appropriate. Has contact with the end-customers.

2. Integration Approach

The overall integration approach (see Figure 1) decouples connectivity between OpenNets CRM Core System and the multiple entities that enable the operational activities necessary to deliver and support end-customer services. The key participants involved in OpenNets services include:

- Service Provider (SP)
- Network Operator (NO)
- Infrastructure Owner (IO)

This multi-party model is underpinned and enabled by OpenNet, acting as both the conduit and integration between the entities. In no situation are the entities expected to have direct contact, rather this is enabled via the OpenNet Integration/System.

The OpenNet CRM & Billing Core System is delivered by OpenNet selected vendor Cerillion. Cerillion additionally provides the Integration Framework to which the OpenNet participants interface, which means that the Network Operator should only communicate with OpenNet Integration Layer.

A Network Operator communicates with the OpenNet Integration Layer over the public IP address via TLS (see Security & Authentication section).

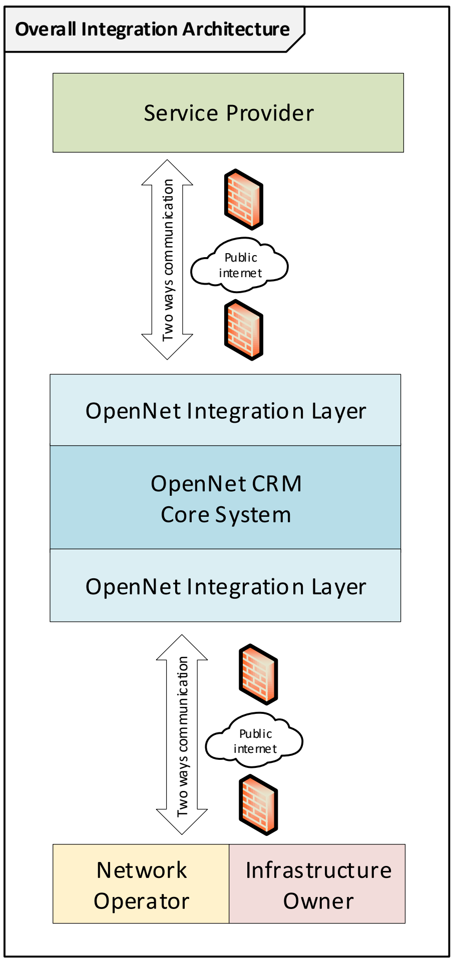


Figure 1: Overall Integration Architecture

3. Integration Framework & Principles

Using Open Source components to enable the Integration Framework that provides a secure, performant and highly available solution, combined with industry standard approaches for integration and enterprise application integration, OpenNet provides a single approach that enables the OpenNet participants to perform specific business functions relevant to the role/action performed in support of the end-to-end solution.

Key principles for the framework include:

- Authentication through username & password or client credentials provide secure access over HTTPs
- Use RESTful interfaces for simple and lightweight payloads
- Provide single patterns for specific interaction
- All participants will consume integration services for get and put operations
- Interfaces will support synchronous and asynchronous using request/response pattern
- Manage all exceptions and error handling gracefully, providing the consumer sufficient information to understand the issue for resolution activities
- Capture application or system exceptions/errors to enable application management

- Use industry standard monitoring to ensure queues, throughput and processing are logged to ensure SLA are supported
- Approach is to provide single services that enable specific business functions
- Exception responses to be dealt with gracefully
- All exceptions and errors to be logged

4. Security & Authentication

OpenNet is going to use the same security model around the OpenNet Integration Layer and Core Systems, which means the same security model applies to the three APIs in the system. To recap these are:

- **OpenNet API:** Published through OpenNet Integration Layer and consumed by the Service Provider
- **Network Operator API:** Published by Network Operator and consumed by OpenNet Integration Layer
- **Infrastructur Owner API:** Published by Infrastructure Owner and consumed by OpenNet Integration Layer

APIs are only available over HTTPs.

Two-Factor Authentication (2FA) is enforced when clients are authenticated. The following factors are used:

- Username and password or client credentials. The username identifies a system user.

The security is strengthened further by establishing IP-address whitelists for each API.

The use of two-way TLS will ensure integrity and confidentiality of the exchanged information and provide a strong authentication of the Network Operator and Infrastructure Owner.

The requirements above will assert a logical separation of Service Provider, Network Operator and Infrastructure Owner for access and data in the OpenNet application, effectively preventing a Service Provider, Network Operator and Infrastructure Owner from accessing any part of data belonging to (or pertaining to business conducted with) another Service Provider, Network Operator and Infrastructure Owner.

The requirements stated above must be adhered to by Cerillion (WCBS) SP and any NO/IO operator. Operators may choose to address the requirements in a number of different ways. It is therefore for important that implementation details are both carefully documented and approved by OpenNet before any production data are exchanged.

4.1 Server Certificates

The three APIs are published on HTTPs URL’s using TLS server certificates. All server certificates used must adhere to current best practices with respect to algorithms, key sizes etc.

API exposed by the OpenNet Integration Layer uses server certificates issued by a CA operated by Cerillion.

NO API and IO API is expected to use TLS certificates issued by a public CA.

4.2 Registration

Before any communication can take place the required credentials must be established at the client systems. This process is described below for each API.

4.2.1 OpenNet API Registration

New Service Providers are registered in the OpenNet Integration Layer, where OpenNet provides the following information to the Service Provider:

- URL of the OpenNet Integration Layer API
- Username and password for Service Provider system account
- CA certificate for Cerillion CA

OpenNet provides the following information to Cerillion:

- Source IP address used by NO/IO

4.2.2 NO/IO Registration

NO/IO systems are also registered in the OpenNet Integration Layer, where NO/IO provides the following information to OpenNet:

- URL of the NO/IO API
- Username and password system account to be used in NO/IO
- CA certificate for used server CA (if not public)

OpenNet provides the following information to NO/IO:

- Source IP address used by OpenNet Integration Layer
- VOCES certificate (issued to OpenNet) used by the OpenNet Integration Layer

OpenNet provides the following information to Cerillion:

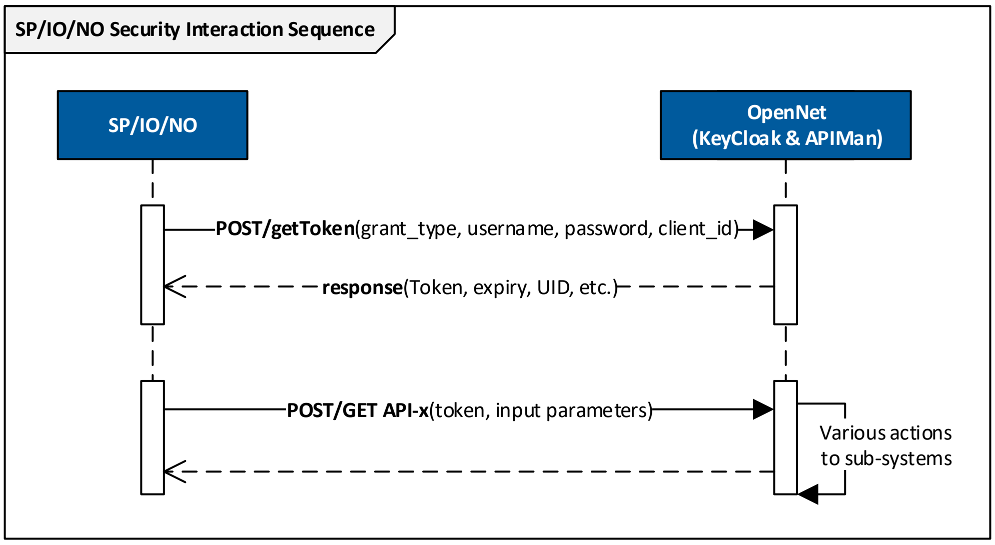
- VOCES certificate and private key for authenticating in NO/IO APIs (or equivalent installation credentials allowing Cerillion to issue VOCES certificate themselves).
- URL of the NO/IO API
- Username and password system accounts to be used in NO/IO
- CA certificate for used server CA (if not public)

4.3 OpenNet Integration Layer Authentication

This section describes the inbound authentication communication when the Network Operator or Infrastructure Owner is going to access OpenNets Integration Layer. OpenNets Integration Layer has chosen to employ an OAuth2 model where one subsystem (KeyCloak) acts as IdP (issuer of OAuth2 tokens) and another subsystem (APIMan) acts as RP (consumer of OAuth2 tokens). An initial request presents username & password or client credentials causing KeyCloak to issue a temporary OAuth2 token.

Subsequent API requests are authenticated exclusively by passing the OAuth2 token in an HTTP header.

Below you can see the sequence diagram for interaction between the Network Operator or Infrastructure Owner.



SP/IO/NO Security Interaction Sequence
SP/IO/NO
OpenNet (KeyCloak & APIMan)
POST/getToken(grant_type, username, password, client_id)
response(Token, expiry, UID, etc.)
POST/GET API-x(token, input parameters)
Various actions to sub-systems

Figure 2: Security Interaction Sequence

The sequence diagram shows the following:

1. The Infrastructure Owner or Network Operator will call a login service with unique username/password. The login is a POST request where two operations will occur:
   - TLS handshake
   - OpenNet Integration Layer which include KeyCloak and APIMan will authenticate and return token if all credential validation rules pass.

2. The token, used in the header for service calls, will allow the Network Operator and Infrastructure Owner the ability to call the desired services (i.e. Service Availability). On the basis that a token is released and the consumer can call the service, all subsequent downstream calls and responses will be managed by OpenNet Integration Layer.

3. OpenNet Integration Layer returns the response to the Infrastructure Owner or Network Operator.

Access tokens is signed and the keys are signed and stored and protected within the KeyCloak server.

1.2.1. Data Definition

The following provides an overview of the data items expected to support the INPUT and RESPONSE payload. This data will be subject to change as design/development evolves through.

Input - Data Item	**Description**
POST getToken	SP will call login service with unique user/password, identifier and type of token required grant_type=xxxxx&client_id=xxxxx&username=xxxxx&password=xxxx
grant_type	Type of request. Available grant types are "password" and "refresh_token"
client_id	Used to identify the client. A client_id of "IL" will automatically create an OAuth Key in the system that is used for "password" authentication
username	The username of the user authenticating to the system
password	The plain text password the user authenticating to the system
Response	
authResponse	"access_token": "xxxx", "expires_in": 299,

	"refresh_expires_in": 1799, "refresh_token": "xxxxx", "token_type": "bearer", "not-before-policy": 0, "session_state": "0fd84d0d-a111-49c2-be01-983561de12c1"
access_token	The OAuth 2.0 access token needed to authenticate for other methods.
expires_in	The length of time until access_token expires in seconds
refresh_expires_in	The length of time until refresh_token expires in seconds
refresh_token	The token needed to extend the access_token expiration timeout
token_type	The token type. Currently only "bearer" is supported
not_before_policy	Not used. Always set to 0
session_state	A unique value that identifies the current user session

Table 2: Data Definition

The access token has the following characteristics:

- The token expire currently as default after 5 minutes
- Any token refresh (issue of new token) requires a new authentication
- The reference specification in the KeyCloak OAuth2 documentation is RFC6749
- The token is validated by APIMan

4.4 NO/IO Authentication

This section describes how OpenNet Integration Layer is authenticated when invoking NO/IO APIs.

As previously stated the same overall security requirements are imposed on NO/IO APIs as on OpenNet API (2-factor authentication using username/password and VOCES).

Instead of adopting an OAuth2 scheme as used by OpenNet Integration Layer it is possible that NO and IO will choose to implement a simpler authentication scheme.

Options include:

- Authenticating every API request fully:
  - Username/password sent as HTTP header
  - Frontend terminates TLS, backend validates VOCES certificate
  - Preferred: Employ a dedicated (account-specific) API-key replacing username/password
  - Login-method invoked with username/password and TLS client cert returns sessionID
    - SessionID authorized subsequent API-invocations
    - SessionID expires after 5 minutes

If no OAuth2/OpenID Connect capabilities are already established at NO/IO the methods above are considerably simpler to implement.

NO and IO must in any case describe how the security requirements are satisfied and document details, including how VOCES certificate validation is performed.

4.5 Logging

The operators (OpenNet, NO, IO) must ensure that the three APIs (OpenNets API, NO's API, IO's API) establish traffic logs.
The traffic logs must include:

- Web Service Calls
- Service-to-Backend Audits
- Request/Response Interactions
- Error and Exceptions

5. Integration Layer Process

The suggested proces for a Network Operator to interact with Wholesale through the Integration Layer require coordination which is described below:

1. Making sure the right resources are allocated
2. Technical onboarding of the Network Operator:
    1. Firewalls, port opening, etc.
    2. Security setup
3. An introductory workshop to include:
    1. An overview of the OpenNet IT interface
    2. Detailed examination of the OpenNet interface documentation
    3. Introduction to the sequence diagrams
    4. Input/output parameters
    5. Error Codes and descriptions
4. Planning of test activities:
    1. Create a testplan
    2. Agree on testcases
    3. Is test data available in the test environment
5. Agreement about support during the development project
6. OpenNet and Network Operator must agree on operational requirements (Incidents, Change, Major Incidents) and support with regard to the OpenNet IT interface

A contact person in OpenNet which will assist and coordinate with the above.

6. Services & Communication

The OpenNet Integration Layer provides all services via RESTful interfaces, delivered over HTTPs and sFTP enables processes that require significant data volume transfer (e.g. address file).

Key consumable services include:

Capability	**Process**	**Type**	**Purpose**
Address File		sFTP	A file which contains addresses with coverage which means the Service Provider can use these Address Files to collect, ingest and use as a part of their sales process. A dedicated secure FTP location accessible through Service Provider specific user/password credentials. The FTP site will primarily be used to collect daily

			supply data files; address coverage, lead time definition etc.
Capacity	Issue_Service_Order /Order Capture	REST	The Network Operator provides a service to check and reserve capacity during the sale process. This feature provides the integration between the OpenNet solution and the network element. The feature includes check and reserve capacity, confirm capacity, cancel and time out capacity.
Get Fulfillment Request	Fulfillment	REST	This service allows NO/IO to get information about orders that have been placed by the SP.
Update Fulfillment Request	Fulfillment	REST	This service allows NO/IO to signal to OpenNet that the specified fulfillment reqest have been completed. Depending on type this can result in a number of different scenarios i.e. progress work to next part of an order, start billing, end billing.
Create Event	Fulfillment	REST	This service allows NO/IO to create events, that is used to notify SP of specific milestones or actions during order fulfillment.
Add Note to order history	Add order note	REST	This service allows NO/IO to add note to an order. The notes are used as order history, and provide a complete view of order status and history.
Get Next Event	N/A	REST	Usability by NO/IO is still pending
Support Tools	Service Assurance	REST	A service which allows the Service Provider with a direct interface to the network layer to retrieve technical data related to the service configuration.
Trouble Ticket	Service Assurance	REST	A service which allows the Service Providers to raise incident and service request tickets, subsequently routed to the appropriate resolver group.
Planned & Unplanned Outages	Service Assurance	REST	A service where information about planned and unplanned outages related to end customers can be submitted by NO's and/or IO's and fetched by SP's. This API is NOT used for ITSM related to the OpenNet IT systems them self.
3rd Party Charging	Billing	sFTP	Enables Network Operator to submit additional charges to wholesale that need to be applied to the Service Provider or additional work-costs incurred during the delivery of the service.
Time Slot Management		REST	This service allows IO to manage technician capacity on a day to day basis.

Table 3: Key Consumable Services Overview

7. Address File

The OpenNet solution will enable large data volume transfer via sFTP and will be used in support of providing Service Providers with address data. The Infrastructure Owner should create and upload a file with an agreement amount of Danish addresses, with coverage, for the Service Provider to collect, ingest and use as part of the sales process. The exact datafields and datainformation is part of the document "Address File".

The sFTP service enables OpenNet and the Service Provider to excange "address file" in a controlled maner. The Service Provider is responsible for accessing the sFTP and retrieving files which is showed in the figure below:

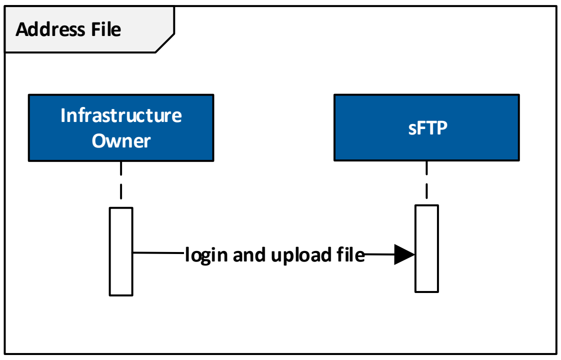


Figure 3: Address File

It needs to be agreed between OpenNet and the Infrastructure Owner at what time of the day the file will be generated and afterwards is available for the Service Provider.

The Infrastructure Owner will be provided access to a dedicated sFTP location, created during the technical onboarding. The sFTP site will be accessed via username & password permissions. Any issues accessing the site or depositing files will be managed by support.

8. Structure & Handling of Work Orders within the OpenNet Solution

When SP places an order to OpenNet, it will result in one or more work orders being created internally in the OpenNet system. These work orders are very simple in their structure, and hold only very little detail about the expected delivery process.

OpenNet handles the orchistration only four simple jobsteps, i.e. a Confirm Capacity job step, a NO work job step, a IO work job step and a final technical provisioning job step. Each of theese job steps result in a fulfillment request being made available to the NO or IO, and they are expected to schedule the needed work in acordance with the delivery date and time information that are provided as part of the fulfillment requests.

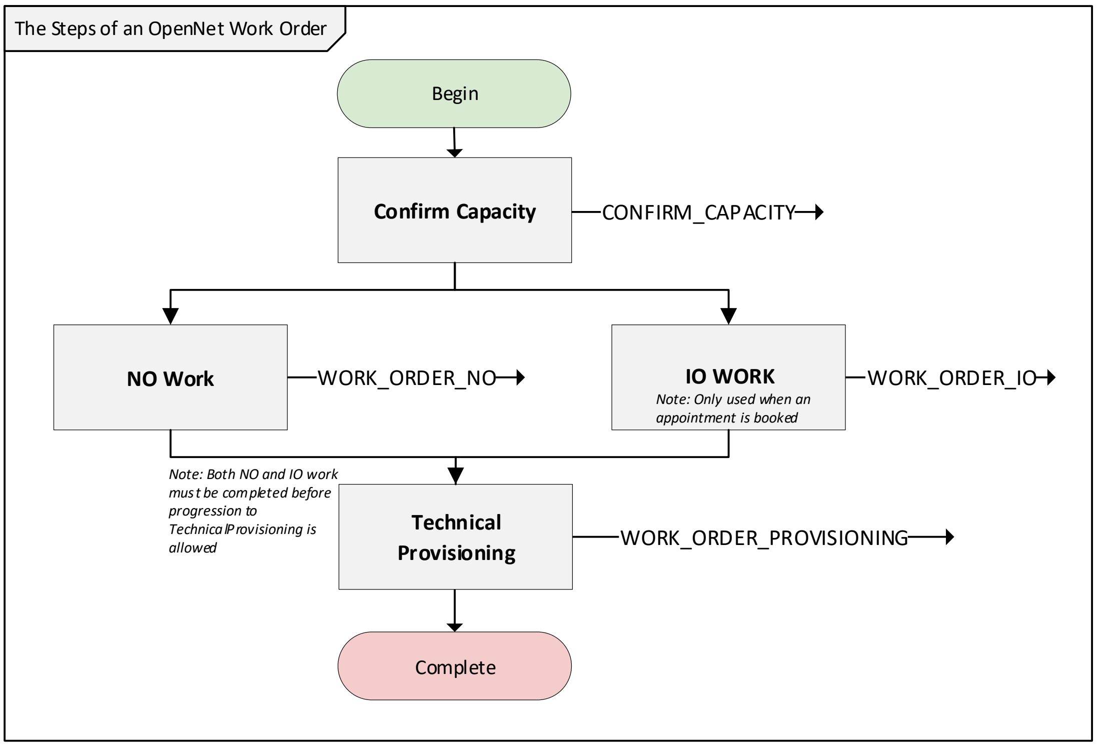

The Steps of an OpenNet Work Order
Begin
Confirm Capacity
CONFIRM_CAPACITY
NO Work
WORK_ORDER_NO
IO WORK
Note: Only used when an appointment is booked
WORK_ORDER_IO
Note: Both NO and IO work must be completed before progression to Technical Provisioning is allowed
Technical Provisioning
WORK_ORDER_PROVISIONING
Complete

Figure 4: The steps of an OpenNet work order

8.1 Handling Order Changes

If an order is changed before it is fully completed, all job steps are reactivated. This means that the NO and IO will receive the fulfillment requests again, and should disregard previously received information. The salesOrderNo, serviceSubscriptionID and workOrderID will remain the same, enabling NO and IO to match the new information to the previously information received. A new requestUID is provided, and the old requestUID will no longer be useable.

8.2 Handling Rescheduling

In case an appointment or action date is rescheduled, the job steps that currently have uncompleted fulfillment requests will be reactivated. This means there corosponding fulfillment request will be reissued with the new information. With the exception of requestUID other ID's passed to NO/IO will remain the same, i.e. serviceSubscriptionID, workOrderID, salesOrderNo etc..

8.3 Handling H1 & H3 Sold Together

In case the SP orders both a H1 and H3 service on the same order, two separate capacityCheckAndReserve requests will be sent to the NO. The two capacityCheckAndReserve requests will hold the same basketID, but other properties e.g. serviceSubscriptionID, productCode and most optionCodes will be different in the two requests.

In case one or both capacityCheckAndReserve requests fail, the complete order will fail in OpenNet and confirmCapacity requests for the two temporary reservations will **not** be sent to NO, unless the issue can be resolved by fixing data inside OpenNet and retry the capacityCheckAndReserve request. In case of a retry, the same basketID will be used.

Unless SP cancels the full order, thereby also sending cancelCapacity for successful reservations, it is expected that NO will trigger CapacityReservationTimeOut requests for any temporary reservation that times out. Minimum reservation time is 1 hour.

In case both capacityCheckAndReserve request succeed, the order will move on in OpenNet where SP will schedule either an action date or appointment. Afterwards two separate work orders will be generated internally, and this will in turn generate multiple south bound fulfillment requests for NO and depending on the order IO. For fulfillment request of type WORK_ORDER_NO and WORK_ORDER_IO and WORK_ORDER_PROVISIONING the same preferedActionDate or appointmentDate and appointmentSlot will be passed for both services. NO/IO is expected to be able to pair the two WORK_ORDER_NO, two WORK_ORDER_IO and two WORK_ORDER_PROVISIONING fulfillment requests together where required using the salesOrderNo property.

If any changes are done to an inflight order, or to an active service, it will always result in a new capacityCheckAndReserve for the specific serviceSubscriptionID. In these requests a new basketID for each change will be introduced, unless the request is a retry due to errors, and the original link between the H1 and H3 services, using the first basketID, is therefore no longer visible in the capacityCheckAndReserve requests.

9. Interaction/Sequence Diagram

This section describes how different parts of the OpenNet Integration Layer performs functions and the order in which the interactions occur when a particular use case/capability is executed. The sequence diagram shows which message/method and primary parameters as unique keys as part for the request and the response interaction.

9.1 Network Capacity Handling

The capacity service is used to first get a temporary reservation of the network capacity for later to confirm the reservation. The capacity will be temporary reserved for a defined period of time only, which is expected to be no less than an hour, it could be any time longer than this. When the reservation of the capacity reservation times out the Network Operator needs to inform OpenNet about the timeout. If an order is cancelled the reserved capacity also needs to be cancelled so the capacity can be released again.

The Capacity service will have the following operations:

- capacityCheck (OpenNet => NO)
- capacityCheckAndReservation (OpenNet => NO)
- confirmCapacityReservation (OpenNet => NO)
- cancelCapacity (OpenNet => NO)
- CapacityReservationTimeOut (NO => OpenNet)

The operation is described in details in the next section.

All calls are expected to be idempotent which means that it will be the same result the capacity service returns.

9.1.1 Capacity Check

The Capacity Check gives the Service Provider the capability to request and get capacity information on a single address towards the Network Operator infrastructure. The response will only reply with capacity information about the end customers installations and will not reserve any capacity.

The sequence diagram for capacity check is showed in the sequence diagram below**:**

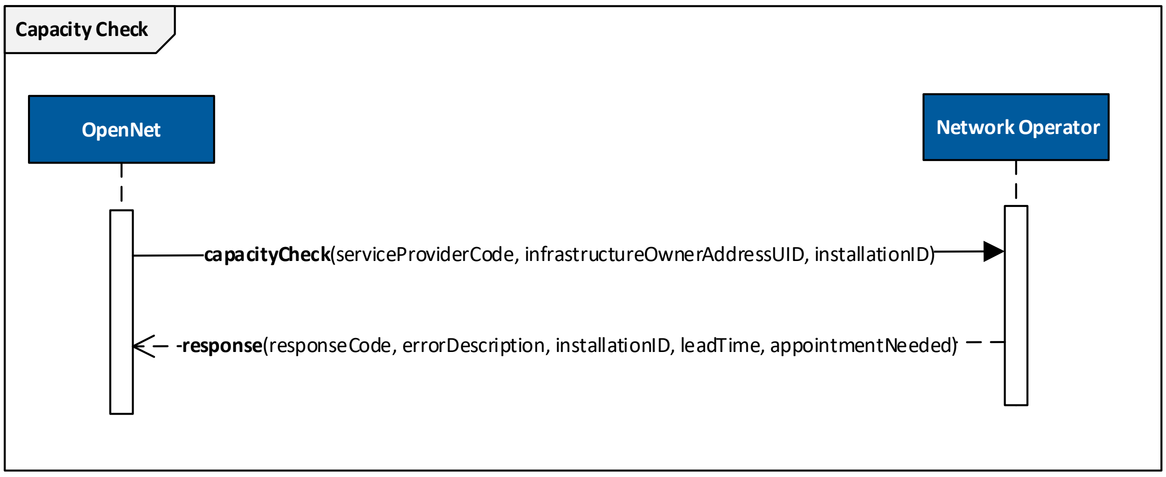

Capacity Check
OpenNet
Network Operator
capacityCheck(serviceProviderCode, infrastructureOwnerAddressUID, installationID)
-response(responseCode, errorDescription, installationID, leadTime, appointmentNeeded)

Figure 5: Capacity Check

9.1.1.1 Data Definition

The following provides a high-level view of the data items expected to support the INPUT and RESPONSE payload. This data will be subject to change as design/development evolves through.

Input - Data Item	**Description**
serviceProviderCode	Service Provider identifier
infrastructureOwnerAddressUID	Unique identifier of the specified address.
technologyType	Fiber, copper, coax
productCode	H1, H2, H3
optionCodes	Multiple codes specifying speed, tv package and other product options. E.g.: Format SPEED100
For each entry for a specific product, parse the following information	
infrastructureOwnerCode	InfrastructureOwnerCode E.g.: 4 character string (e.g. IO_E)
installationID	Unique identifier for the installation. This value can be empty string/null if no installation has been supplied at the end customer address.
networkOperatorDataField1	Data stored for specified Installation id and product. (Provided by IO via supply data) ,
networkOperatorDataField2	Data stored for specified Installation id and product. (Provided by IO via supply data)
networkOperatorDataField3	Data stored for specified Installation id and product. (Provided by IO via supply data)
networkOperatorDataField4	Data stored for specified Installation id and product. (Provided by IO via supply data)
networkOperatorDataField5	Data stored for specified Installation id and product. (Provided by IO via supply data)
networkOperatorDataField6	Data stored for specified Installation id and product. (Provided by IO via supply data)
Response	
responseCode	0: Success 1: No Available Capacity 2: Other Error
message	String specifying cause of error when present.

	If responseCode is 1 this property will hold a JSON encoded string in the following format: ”{\”reasonCode\”:integer,\”description\”:\”text describing scenario\”}” The reasonCode is an integer defined by OpenNet for each relevant scenario. The description is also a fixed text defined by OpenNet. **Note: For details on defined reason codes and descriptions please see the OpenNet Appendix.**
installationsID	InstallationID on which the service will be delivered. If no installationID was supplied during the request, this Installation_id would have been generated by the NO. It is considered best practice to keep the installationID returned static per address/installation even if an installation does not yet exist on the address, as the SP will then see the same installation ID if they call the API multiple times, as well as well as when the SP submits an order later in the process.
leadTime	Lead time for NO to make capacity available. Lead time will be a number of calendar days
appointmentNeeded	"N": No need for IO technician appointment on delivery date. "Y": Need for IO technician appointment on delivery date.
validCPEFeature	**H1:** Gives information about the end customers installation and og how the installation can support the requested products primary regarding speed. For H1 this string is expected to hold the maximum supported speed of the installation after potential fiberbox change. E.g. "SPEED1000" **H2 and H3:** Return null or empty string “”

Table 4: Data Definition

The same data definition is described in the .json files in the table below. The files describe the structure when OpenNets Integration Layer is sending the files to the Network Operator.

Description	**JSON**

Request	{
	
	  "serviceProviderCode": 20000001,
	
	  "infrastructureOwnerAddressUID": "99999999-9999-9999-999999991235",
	
	  "technologyType": "FIBRE",
	
	  "productCode": "H1",
	
	  "OptionCodes": [
	
	    "SPEED120"
	
	  ],
	
	  "supplyData": [

```
    {
      "infrastructureOwnerCode": "IO01",
      "installationID": "7777",
      "networkOperatorDataField1": "string1",
      "networkOperatorDataField2": "string2",
      "networkOperatorDataField3": "string3",
      "networkOperatorDataField4": "string4",
      "networkOperatorDataField5": "string5",
      "networkOperatorDataField6": "string6"
    },
    {
      "infrastructureOwnerCode": "IO01",
      "installationID": "8888",
      "networkOperatorDataField1": "string1",
      "networkOperatorDataField2": "string2",
      "networkOperatorDataField3": "string3",
      "networkOperatorDataField4": "string4",
      "networkOperatorDataField5": "string5",
      "networkOperatorDataField6": "string6"
    }
  ]
}
```

Response

```
{
  "responseCode": 0,
  "message": "",
  "installationID": "7777",
  "leadTime": 11,
  "appointmentNeeded": "Y",
```

	"validCPEFeatures": "string", }

Table 5: Data Definition in JSON format

9.1.2 Capacity Check & Reservation

The Capacity Check and Reservation service is used to check and reserve capacity during the sales process. The service is activated by the OpenNet Core System by the Integration Layer and will be communicated to the Network Operator for further processing.

The sequence diagram for Check Capacity and Reservation is showed below:

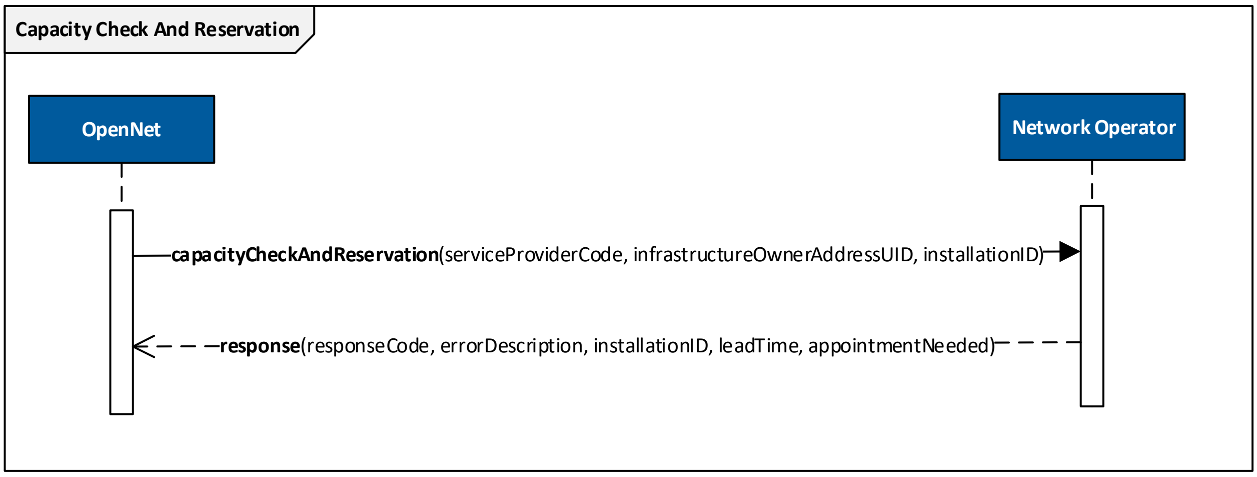

Capacity Check And Reservation
OpenNet
Network Operator
capacityCheckAndReservation(serviceProviderCode, infrastructureOwnerAddressUID, installationID)
response(responseCode, errorDescription, installationID, leadTime, appointmentNeeded)

Figure 6: Capacity Check & Reservation

9.1.2.1 Data Definition

The following provides a high-level view of the data items expected to support the INPUT and RESPONSE payload. This data will be subject to change as design/development evolves through.

Input - Data Item	**Description**
serviceProviderCode	Service Provider identifier
serviceSubscriptionID	Unique identifier of the main service for which capacity is reserved. This unique identifier is constant across order fulfillment process and stays the same, even during a possible option change before the original order has been fulfilled. E.g.: Format is 10 digits
basketID	A unique identifier of the order requested from the Service Provider. The unique identifier id is used to cancellation i relation to and/or work order(s) if reservation time runs out within NO.
infrastructureOwner AddressUID	Unique identifier of the specified address
technologyType	FIBRE, COAX, COPPER
productCode	H1, H2, H3

optionCodes	Multiple codes specifying speed, tv package and other product options. E.g.: Format Speed100
serviceProviderIn1 serviceProviderIn2 serviceProviderIn3 serviceProviderIn4 serviceProviderIn5 serviceProviderIn6	Theese fields are used to parse information of a more technical nature i.e. VLAN ID's, poiID etc. The below information is what should be expected by the Network Owner. **Note**: If a specific installationID is required to be used on the endcustomer address, the requiredInstallationID property needs to be provided. If no required installationID is provided, the network owner will select the most suitable installation to use in case multiple installations exist on the address. If the SP passed a requiredInstallationID which does not match any of the array elements passed in the supplyData array in the request, NO is expected to return an error. If the NO has opted to add the needed logic, which is not required by OpenNet, a success response can be returned if the required installationID is linked to the specific address but has not yet uploaded to OpenNet. **H1** **serviceProviderIn1 :** “{\"cvlan\":[{\"id\":32,\"tagged\":true},{\"id\":33,\"tagged\":true},{\"id\":34,\"tagged\":true},{\"id\":35,\"tagged\":true},{\"id\":36,\"tagged\":true},{\"id\":37,\"tagged\":true},{\"id\":38,\"tagged\":true},{\"id\":null,\"tagged\":false}]}” **serviceProviderIn2 :** null or ”{\"multicastCvlan\":{\"id\":35,\"tagged\":true}}” or ”{\"multicastCvlan\":{\"id\":null,\"tagged\":false}}” or ”{\"multicastCvlan\":{\"id\":null,\"tagged\":null}}” **serviceProviderIn3 :** null or ”{\”requiredInstallationID\”:\”BA9382XXX\”}” or ”{\”requiredInstallationID\”:null}” **serviceProviderIn4-6 :** null or ”” **H2** **serviceProviderIn1 :** "{\”poiID\":\"JDMSK78986546\"}" **serviceProviderIn2 :** null or "{\"requiredInstallationID\": \"BCJD8234234\"}" or "{\"requiredInstallationID\": null}" **serviceProviderIn3-6 :** null or ”” **H3** **serviceProviderIn1 :** null or ”{\”requiredInstallationID\”:\”BA9382XXX\”} or ”{\”requiredInstallationID\”:null} **serviceProviderIn2-6 :** null or ””**serviceProviderIn2-6 :** null or ””
For each entry in supply data for specific product, parse the following information	
infrastructureOwnerCode	InfrastructureOwnerCode E.g.: 4 character string (e.g. IO_E)
installationID	Unique identifier for the installation. This value can be empty string/null if no installation has been supplied at the end customer address.
networkOperatorDataField1	Data stored for specified Installation id and product. (Provided by IO via supply data)
networkOperatorDataField2	Data stored for specified Installation id and product. (Provided by IO via supply data)
networkOperatorDataField3	Data stored for specified Installation id and product. (Provided by IO via supply data)
networkOperatorDataField4	Data stored for specified Installation id and product. (Provided by IO via supply data)
networkOperatorDataField5	Data stored for specified Installation id and product. (Provided by IO via supply data)
networkOperatorDataField6	Data stored for specified Installation id and product. (Provided by IO via supply data)
Response	
responseCode	0: Success 1: No Available Capacity 2: Other Error
message	String specifying cause of error when present.

	If responseCode is 1 this property will hold a JSON encoded string in the following format: ”{\”reasonCode\”:integer,\”description\”:\”text describing scenario\”}” The reasonCode is an integer defined by OpenNet for each relevant scenario. The description is also a fixed text defined by OpenNet. **Note: For details on defined reason codes and descriptions please see the OpenNet Appendix.**
installationsID	InstallationID on which the service will be delivered. If no installationID was supplied during the request, this Installation_id would have been generated by the NO. InstallationID would need to be stored on the service
leadTime	Lead time for NO to make capacity available. Lead time will be a number of calendar days
appointmentNeeded	"N": No need for IO technician appointment on delivery date. "Y": Need for IO technician appointment on delivery date.
networkOperatorOut1	Information should be parsed to NO as part of fulfillment requests.
networkOperatorOut2	Information should be parsed to NO as part of fulfillment requests.
networkOperatorOut3	Information should be parsed to NO as part of fulfillment requests.
networkOperatorOut4	Information should be parsed to NO as part of fulfillment requests.
networkOperatorOut5	Information should be parsed to NO as part of fulfillment requests.
networkOperatorOut6	Information should be parsed to NO as part of fulfillment requests.
serviceProviderOut1 serviceProviderOut2 serviceProviderOut3 serviceProviderOut4 serviceProviderOut5 serviceProviderOut6	The technical provisioning data: **<u>H1</u>** **serviceProviderOut1 :** “{\“port\“:\“ethPort1\“}“ **serviceProviderOut2 :** “{\"tvlan\":[{\"id\":4001},{\"id\":4002},{\"id\":4003},{\"id\":4004},{\"id\":4005},{\"id\":4006},{\"id\":4007},{\"id\":4008}]}” **serviceProviderOut3 :** ”{\”svlan\”:234,\”poiID\”:\”GFS789XXX\”}” **serviceProviderOut4-6 :** null or ”” **<u>H2</u>** **serviceProviderOut1 :** ”{\”transceiverType\”:\”single\”}” or ”{\”transceiverType\”:\”dual\”}” **serviceProviderOut2 :** ”{\”svlan\”:234}” **serviceProviderOut3 :** ”{\”transceiverSpeed\”:\”1G\”}” or ”{\”transceiverSpeed\”:\”10G\”}” **serviceProviderOut4-6 :** null or ”” **<u>H3</u>** **serviceProviderOut1 :** “{\“port\“:\“rfPort\“}“ **serviceProviderOut2-6 :** null or ”” **Note**: As a minimum the serviceProviderOut1 field must be populated for H1, H2 and H3 services in the capacityCheckAndReserve response. For H2 also the transceiverSpeed in serviceProviderOut3 is expected to be returned as part of the capacityCheckAndReserve Response. Other values are allowed not to be published until the actual order fulfillment process, but must be populated no later than the completion of the WORK_ORDER_NO fulfillment request. Maximum length of UDF field is 256 characters. The returned values for T-VLAN’s must one to one map to the inputted C-VLAN’s, i.e. first T-VLAN id will map to first C-VLAN id.

InfrastructureOwnerOut1	These fields will hold information needed by IO during order fulfillment. Information should be parsed to IO as part of fulfillment/provisioning messages.
InfrastructureOwnerOut2	These fields will hold information needed by IO during order fulfillment. Information should be parsed to IO as part of fulfillment/provisioning messages.
InfrastructureOwnerOut3	These fields will hold information needed by IO during order fulfillment. Information should be parsed to IO as part of fulfillment/provisioning messages.
InfrastructureOwnerOut4	These fields will hold information needed by IO during order fulfillment. Information should be parsed to IO as part of fulfillment/provisioning messages.
InfrastructureOwnerOut5	These fields will hold information needed by IO during order fulfillment. Information should be parsed to IO as part of fulfillment/provisioning messages.
InfrastructureOwnerOut6	These fields will hold information needed by IO during order fulfillment. Information should be parsed to IO as part of fulfillment/provisioning messages.

Table 5: Data Definition

The same data definition is described in the .json files in the table below. The files describe the structure when OpenNets Integration Layer is sending the files to the Network Operator.

Description	**JSON**

Request	{
	
	  "serviceProviderCode": 20000001,
	
	  "serviceSubscriptionID": "570874",
	
	  "basketID": 75646,
	
	  "infrastructureOwnerAddressUID": "99999999-9999-9999-999999991235",
	
	  "technologyType": "FIBRE",
	
	  "productCode": "H1",
	
	  "OptionCodes": [
	
	    "SPEED120",
	
	    "SLASILVER"
	
	  ],
	
	  "serviceProviderIn1": "string1",
	
	  "serviceProviderIn2": "string2",
	
	  "serviceProviderIn3": "string3",
	
	  "serviceProviderIn4": "string4",

	"serviceProviderIn5": "string5",
	"serviceProviderIn6": "string6"
	"supplyData": [
	{
	"infrastructureOwnerCode": "IO01",
	"installationID": "7777",
	"networkOperatorDataField1": "string1",
	"networkOperatorDataField2": "string2",
	"networkOperatorDataField3": "string3",
	"networkOperatorDataField4": "string4",
	"networkOperatorDataField5": "string5",
	"networkOperatorDataField6": "string6"
	},
	{
	"infrastructureOwnerCode": "IO01",
	"installationID": "8888",
	"networkOperatorDataField1": "string1",
	"networkOperatorDataField2": "string2",
	"networkOperatorDataField3": "string3",
	"networkOperatorDataField4": "string4",
	"networkOperatorDataField5": "string5",
	"networkOperatorDataField6": "string6"
	}
	]
	}

Response	{ "responseCode": 0,

	"message": "",
	"installationID": "7777",
	"leadTime": 11,
	"appointmentNeeded": "Y",
	"networkOperatorOut1": "string1",
	"networkOperatorOut2": "string2",
	"networkOperatorOut3": "string3",
	"networkOperatorOut4": "string4",
	"networkOperatorOut5": "string5",
	"networkOperatorOut6": "string6",
	"serviceProviderOut1": "string1",
	"serviceProviderOut2": "string2",
	"serviceProviderOut3": "string3",
	"serviceProviderOut4": "string4",
	"serviceProviderOut5": "string5",
	"serviceProviderOut6": "string6",
	"infrastructureOwnerOut1": "string1",
	"infrastructureOwnerOut2": "string2",
	"infrastructureOwnerOut3": "string3",
	"infrastructureOwnerOut4": "string4",
	"infrastructureOwnerOut5": "string5",
	"infrastructureOwnerOut6": "string6"
	}

Table 6: Data Definition JSON format

9.1.3 Confirm Capacity Reservation

Once an appointment has been scheduled the OpenNet solution will invoke a request to confirm a previously reserved capacity to ensure the service can be delivered/activated on the network. Using the serviceSubscriptionID the confirm capacity service will invoke a call to the Network Operator to confirm capacity reservation, using the same data items included in the original check/reservation request

It is critical that the operation performed by this API is kept to a minimum, as returning an error will result in manual handling and potential order failure. The expected action performed by the API is to stop the temporary capacity reservation from timing out. Any product validation, and other actions that could fail the order in the first place, is expected to have been done in the Capacity Check and Reserve API it self, as there is a clear communication path back to the SP in that process.

An error is only expected to be returned if the Network Operator systems, needed to process the API request, is down, or Network Operator and OpenNet systems has gotten out of sync due to an incident.

The sequence diagram for confirm capacity reservation is showed below**:**

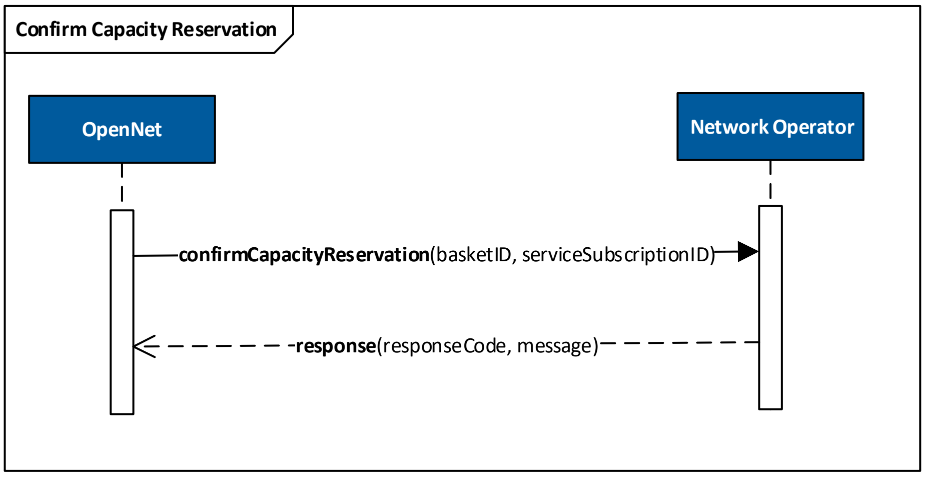

Confirm Capacity Reservation
OpenNet
Network Operator
confirmCapacityReservation(basketID, serviceSubscriptionID)
response(responseCode, message)

Figure 7: Confirm Capacity Reservation

9.1.3.1 Data Definition

The following provides a high-level view of the data items expected to support her INPUT and RESPONSE payload. This data is will be subject to change as design/development evolves through.

Input - Data Item	**Description**
serviceProviderCode	Service Provider identifier
serviceSubscriptionID	Unique identifier of the main service for which capacity is reserved. This unique identifier is constant across order fulfillment process and stays the same, even during a possible option change before the original order has been fulfilled. E.g.: Format is 10 digits
basketID	A unique identifier of the order requested from the Service Provider. The unique identifier ID is used to cancellation i relation to and/or work order(s) if reservation time runs out within NO.
infrastructureOwnerAddressUID	Unique identifier of the specified address.
technologyType	FIBRE, COAX, COPPER

productCode	H1, H2, H3
optionCodes	Multiple codes specifying speed, tv package and other product options. E.g.: Format SPEED100
serviceProviderIn1	Theese fields are used to parse information of a more technical nature i.e. VLAN ID's, poiID etc. The below information is what should be expected by the Network Owner. **Note**: If a specific installationID is required to be used on the end-customer address, the requiredInstallationID property needs to be provided. If no required installationID is provided, the network owner will select the most suitable installation to use in case multiple installations exist on the address. **<u>H1</u>** **serviceProviderIn1 :** “{"cvlan":[{"id":32,"tagged":true},{"id":33,"tagged":true},{"id":34,"tagged":true},{"id":35,"tagged":true},{"id":36,"tagged":true},{"id":37,"tagged":true},{"id":38,"tagged":true},{"id":null,"tagged":false}]}” **serviceProviderIn2 :** null eller ”{"multicastCvlan":{"id":35,"tagged":true}}” eller ”{"multicastCvlan":{"id":null,"tagged":false}}” eller ”{"multicastCvlan":{"id":null,"tagged":null}}” **serviceProviderIn3 :** null or ”{”requiredInstallationID”:”BA9382XXX”}” or ”{”requiredInstallationID”:null}” **serviceProviderIn4-6 :** null or ”” **<u>H2</u>** **serviceProviderIn1 :** "{"poiID":"JDMSK78986546"}" **serviceProviderIn2 :** null or "{"requiredInstallationID": "BCJD8234234"}" or "{"requiredInstallationID": "null"}" **serviceProviderIn3-6 :** null or ”” **<u>H3</u>** **serviceProviderIn1 :** null or ”{”requiredInstallationID”:”BA9382XXX”} or ”{”requiredInstallationID”:”null”} **serviceProviderIn2-6 :** null or ””
serviceProviderIn2	
serviceProviderIn3	
serviceProviderIn4	
serviceProviderIn5	
serviceProviderIn6	
infrastructureOwnerCode	InfrastructureOwnerCode E.g.: 4 character string (e.g. IO_E)
installationID	Unique identifier for the installation. This value can be empty string/null if no installation has been supplied at the end customer address.
networkOperatorOut1	Pupulated by NO during capacityCheckAndReserve call.
networkOperatorOut2	
networkOperatorOut3	
networkOperatorOut4	
networkOperatorOut5	
networkOperatorOut6	
Response	

responseCode	0: Success 1: No Capacity Available (Not used here) 2: Other Error
message	String specifying cause of error when present.

Table 7: Data definition

The same data definition is described in the .json files in the table below. The files describe the structure when OpenNets Integration Layer is sending the files to the Network Operator.

Description	JSON
Request	
Response	

Table 8: Data definition .json

9.1.4 Cancel Capacity Reservation

When orders are cancelled (e.g. prior to activation) or services are ceased, network capacity must be released to ensure the Network Operator has an accurate view of the network (technical) use.

Cancelling/Cease activities will subsequently cancel capacity use on the network.

Using the serviceSubscriptionID only, the cancel capacity service will invoke a call to the Network Operator to release capacity.

The sequence diagram for Cancel Capacity Reservation is showed below:

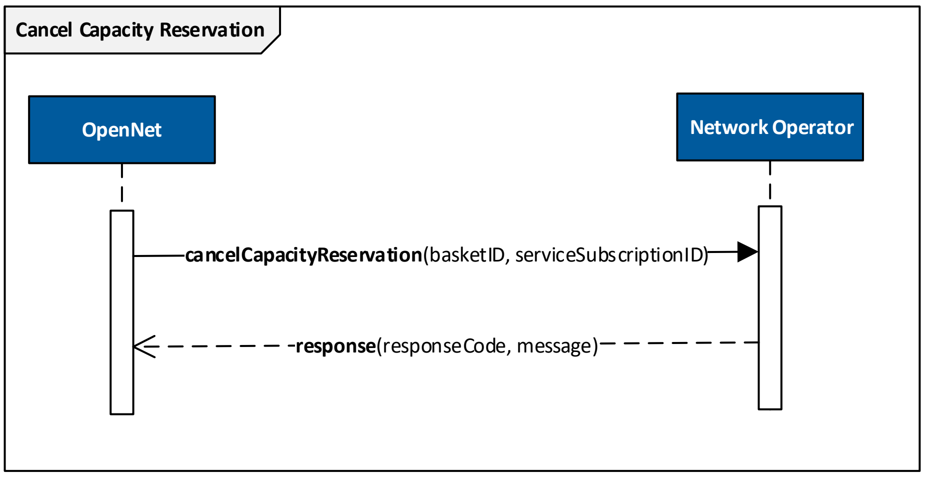

Cancel Capacity Reservation
OpenNet
Network Operator
cancelCapacityReservation(basketID, serviceSubscriptionID)
response(responseCode, message)

Figure 8: Cancel Capacity Reservation

9.1.4.1 Data Definition

Input - Data Item	Description
serviceSubscriptionID	Unique identifier of the main service for which capacity is reserved. This unique identifier is constant across order fulfillment process and remains the same, even during a possible option change before the original order has been fulfilled.

	E.g.: Format is 10 digits
basketID	A unique identifier of the order requested from the Service Provider. The unique identifier ID is used for cancellation in relation to and/or work order(s) if reservation time runs out within NO.
Response	
responseCode	0: Success 1: No Available Capacity 2: Other Error
message	String specifying cause of error when present.

Table 9: Data Definition

The same data definition is described in the .json files in the table below. The files describe the structure when OpenNets Integration Layer is sending the files to the Network Operator.

Description	**JSON**
Request	
Response	

Table 9: Data Definition

9.1.5 Capacity Reservation Timeout

To perform cancel order in case the temporary capacity reservation times out. It's only allowed to remove the temporary reservation if a success response is received from OpenNet – the infrastructure owner is required to hold the reservation in any case of an unsuccessfull response.

The sequence diagram for time out regarding capacity reservation is showed below:

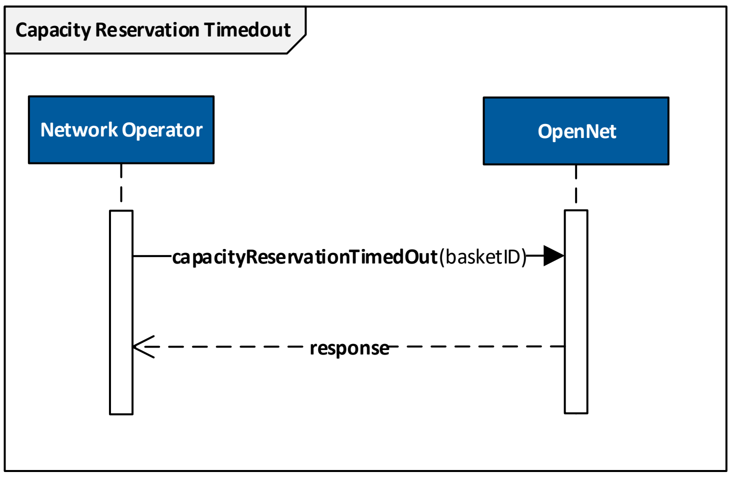

Capacity Reservation Timedout
Network Operator
OpenNet
capacityReservationTimedOut(basketID)
response

Figure 9: Capacity Reservation Timeout

9.1.5.1 Data Definition

Input - Data Item	**Description**
basketID	basketID received by NO as part of the Capacity Check And Reserve request. The unique identifier ID is used for cancellation in relation to and/or work order(s) if reservation time runs out within NO.

(Query parameter)	
Response	
http response 204	Basket and related work orders have been cancelled successfully in OpenNet.
http response 4xx/5xx	Basket and related work orders have not been cancelled. Do not release the temporary reservation.

Table 10: Data definition

10. Order fulfilment

10.1 Fulfilment requests

The Fulfillment Request service is used by NO/IO to collect fulfillment requests and after completion of the specified work at NO/IO, the service is then used to signal OpenNet that the task has been done.

The service will have the following operations:

- getFulfillementRequest
- updateFulfillmentRequest

The operations are described in more details in the next sections.

10.1.1 Get Fulfilment Request

Used by NO/IO to collect fulfillment requests from OpenNet.
The sequence diagram for getFulfillmentRequest is showed below:

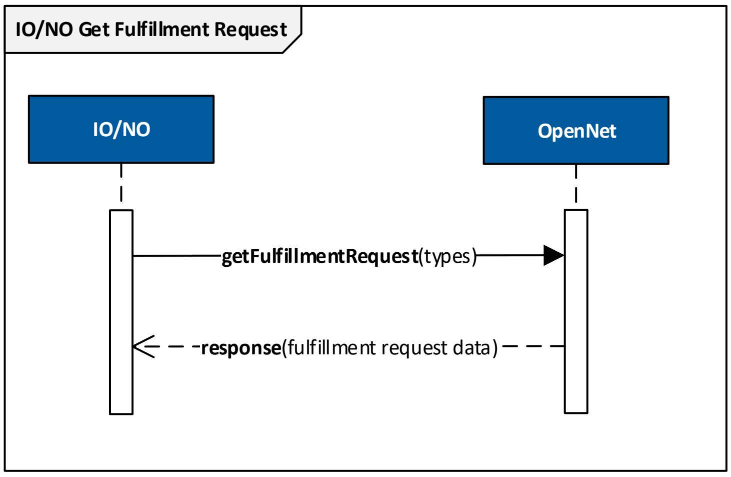

IO/NO Get Fulfillment Request
IO/NO
OpenNet
getFulfillmentRequest(types)
response(fulfillment request data)

Figure 10: Get Fulfillment Request

10.1.1.1 Data Definition

Input - Data Item	**Description**

autoAcknowledge (Optional)	Data type: Boolean Default value is true if not specified, i.e. the returned fulfillment request is removed from the queue once fetched. If set to false, fulfillment request is not removed from the queue until NO/IO actively update the status of the specific fulfillment request. See "Update Fulfillment Request" for details on how to acknowledge a fulfillment request.

requestType	Array of request types to look for. Possible types are: WORK_ORDER_NO, WORK_ORDER_IO, WORK_ORDER_PROVISIONING, WORK_ORDER_TERMINATE, WORK_ORDER_CANCELLED_NO, WORK_ORDER_CANCELLED_IO, WORK_ORDER_CANCELLED_PROVISIONING, BARRING, UNBARRING.
	
	<u>Function change from version 3 onwards:</u>
	OpenNet system will search the internal database holding all fulfillment requests, for the oldest uncollected fulfillment request available to the calling entity. Filtration will ensure that only specified requestType's are considered in the search, and the functional change is that they will be considered equal. In other words, the fulfillment request queue will follow a FIFO principle.
	
	<u>Prior versions:</u>
	OpenNet will try the return a fulfillment request of the first requestType specified. If no fulfillment requests of the first requestType is available, then OpenNet will move to the next specified requestType and search for a fulfillment request of that type. If no fulfillment request of the second request type is found, then OpenNet will move on to the third requestType and so on.
	
	Request example for a typical NO entity:
	{
	  "requestType":[
	    "WORK_ORDER_NO",
	    "WORK_ORDER_PROVISIONING",
	    "WORK_ORDER_CANCELLED_NO",
	    "WORK_ORDER_CANCELLED_PROVISIONING",
	    "WORK_ORDER_TERMINATE",
	    "BARRING",
	    "UNBARRING"
	  ]
	}

Response	
Response (http response code 200)	Request was successful and a fulfillment request has been returned in the response body
requestType	The actual type of request returned.
Other data fields	The actual data of the request. See response data content below.
Response empty response	Desscription
http response code 204	No fulfillment requests of the specified types are available.

Table 11: Data definition

10.1.1.2 GetFulfilmentRequest Response Definition

Field name	**Description**	**Included in Type 1**	**Included in Type 2**	**Included in Type 3**	**Included in Type 4**
requestType	Type can be used to route requests within receiving system. Possible values:	YES	YES	YES	YES

	Type 1: WORK_ORDER_NO, WORK_ORDER_IO, WORK_ORDER_PROVISIONING Type 2: WORK_ORDER_CANCELLED_NO, WORK_ORDER_CANCELLED_IO, WORK_ORDER_CANCELLED_PROVISIONING **Note**: Type 2 fulfillment requests are for information only. This type of fulfillment request should not be completed by NO/IO. Type 3: WORK_ORDER_TERMINATE Type 4: BARRING, UNBARRING **Note**: If a barred service is changed by the SP, no UNBARRING fulfillment request will be generated and expectation is that the service is active once Change of Options order completes. If Change of Options order is cancelled prior to order completion, the service is expected to stay barred.				
requestTypeSequenceNo	Sequence number available for each request type. This enables NO/IO to detect if a request is lost.	YES	YES	YES	YES
requestUID	ID within Cerillion Core, to enable NO and IO to complete the related job step.	YES	YES	YES	YES
generatedDateTime	Date and time, when request was generated.	YES	YES	YES	YES
salesOrderNo	ID within Cerillion Core, that is generated when a sale is committed. This ID can be the same for multiple work orders. (H1 and H3 are sold together)	YES	NO	NO	NO
workOrderID	ID within Cerillion core.	YES	YES	YES	NO
basketID	ID within Cerillion Core. (Maximum is 11 characters)	YES	NO	NO	NO
serviceProviderCode	SP account number. (Format is 8 digit: E.g. 20071234)	YES	NO	NO	NO
serviceSubscriptionID	ID of the main service. This ID is the constant across order fulfillment process and stays the same, even during a possible option change before the original order has been fulfilled. (Format is 10 digits)	YES	YES	YES	YES
infrastructureOwnerCode	Needed by NO, in case NO is operator for more than one IO. 4 character string E.g. IO_E	YES	NO	NO	NO

infrastructureOwnerAddressUID	The address uid of the specified address, as written by IO in supply data.	YES	NO	NO	NO
installationID	ID owned by NO/IO.	YES	NO	NO	NO
technologyCode	FIBRE, COPPER, COAX	YES	NO	NO	NO
productCode	H1, H2, H3	YES	NO	NO	NO
optionCodes	Multiple codes specifying speed, TV package and other product options.E.g. SPEED100	YES	NO	NO	NO
appointmentDate	If an appointment has been scheduled, this field will hold the appointment date. If no appointment is scheduled this expected to be NULL.	YES	NO	NO	NO
appointmentSlot	If an appointment has been scheduled, this field will hold the appointment slot identifier. Slots are expected to cover 8-12, 12-16, 8-10, 12-14 or similar. If no appointment is scheduled this expected to be NULL. Normal early slot: AM_NO Premium early slot: AM_PR_1/AM_PR_2 Normal late slot: PM_NO Premium late slot: PM_PR_1, PM_PR_2 and PM_PR_3 Legacy codes: AM_PR and PM_PR may exist for some IO's.	YES	NO	NO	NO
preferedActionDate	If no appointment is required for order fulfillment, this field will hold the requested activation date. Otherwise it will be NULL.	YES	NO	YES	NO
contactEmail **Available from API version 2.0**	End-customer contact e-mail. Maximum length is 200 characters.	YES	NO	NO	NO
contactName	End-customer contact name. Maximum length 40 characters.	YES	NO	NO	NO
contactPhoneNumber	End-customer contact phone number. Maximum length 15 characters.	YES	NO	NO	NO
priority **Available from API version 3.0**	Value is a Boolean. If this value is true, the appointment or actionDate was scheduled using priority=true from the SP, i.e. the order has been agreed between SP and IO to be fast tracked.	YES	NO	YES	NO
serviceProviderIn1 **serviceProviderIn2** **serviceProviderIn3** **serviceProviderIn4** **serviceProviderIn5**	UDF's populated by SP during order creation.	YES	NO	NO	NO

serviceProviderIn6	For SP this will be used e.g. when placing an order for a H2 service, where SP can select POI id when placing the order.				
networkOperatorOut1	UDF's populated by Capacity check and reserve response, and most likely manipulated during order fulfillment by IO to prepare correct provisioning data. E.g. add CPE serial number to UDF.	YES	NO	NO	NO
networkOperatorOut2					
networkOperatorOut3					
networkOperatorOut4					
networkOperatorOut5					
networkOperatorOut6					
InfrastructureOwnerOut1	UDF's populated by Capacity check and reserve response.	YES	NO	NO	NO
InfrastructureOwnerOut2					
InfrastructureOwnerOut3					
InfrastructureOwnerOut4					
InfrastructureOwnerOut5					
InfrastructureOwnerOut6					

Table 12: Data definition - Fields in a fulfillment request

Description	**JSON**

Request	{
	  "requestType":[
	    "WORK_ORDER_NO",
	    "WORK_ORDER_PROVISIONING",
	    "WORK_ORDER_CANCELLED_NO",
	    "WORK_ORDER_CANCELLED_PROVISIONING",
	    "WORK_ORDER_TERMINATE",
	    "BARRING",
	    "UNBARRING"
	  ]
	}

Table 13: Data Definition JSON format

10.1.2 Update Fulfillment Request

Used by NO/IO to complete fulfillment requests received from OpenNet.

If NO/IO has opted to use the autoAcknowledge=false feature when collecting a fulfillment request, this service is also used to acknowledge reception by updating the status to COLLECTED.

The sequence diagramcfor updateFulfillmentRequest is showed below:

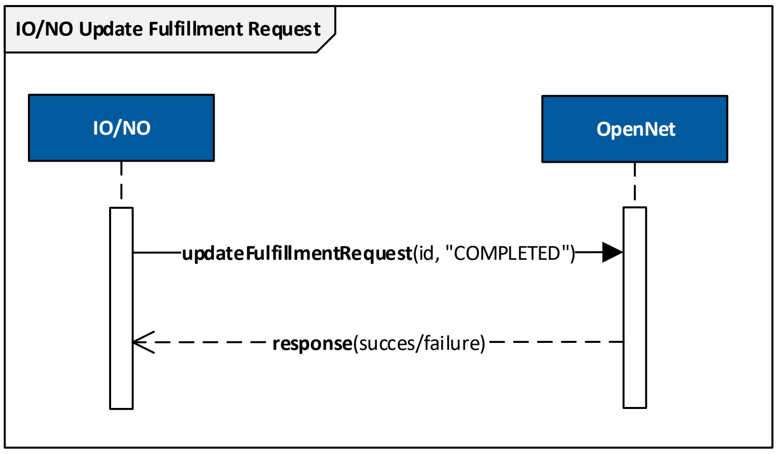

IO/NO Update Fulfillment Request
IO/NO
OpenNet
updateFulfillmentRequest(id, "COMPLETED")
response(succes/failure)

Figure 11: Update Fulfillment Request

10.1.2.1 Data Definition

Input - Data Item	**Description**
id	requestUID from the request that needs to be completed. Cancel requests are auto completed, and has no requestUID for that reason.
status	To signal that a task communicated via a fulfillment request has been completed, set this parameter to "COMPLETED". To acknowledge a fulfillment request, which has been collected with autoAcknowledge=false, set this parameter to "COLLECTED".
Response (http response code 204)	Description
NULL	Fulfillment request completed successfully
Response **(http response code 400)**	Description
responseCode	"FAILURE"
message	"what went wrong"
Response (http response code 500)	
NULL	Fulfillment request could not be completed due to internal server error

Table 14: Data definition

10.2 Add Notes

The notes service enables NO/IO to add notes to orders. The notes will function as the order history.
The sequence diagram for addNote is showed below:

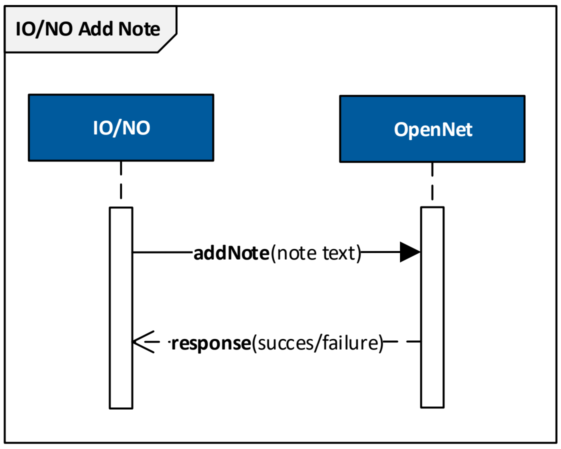


Figure 12: Add Note

10.2.1 Data Definition

Input - Data Item	**Description**
serviceSubscriptionID	
id	basketID
noteType	A categorization that enables SP to filter general notes from machine readable notes. Expected types are NONOTE, IONOTE, GENOTE.

note	**GENOTE**: General note
	This is a just text in unstructured format, that will be made available to the Service Provider.
	
	IONOTE/NONOTE: Infratructure Owner Note/Network Operator Note
	These are structured notes, that are machine readable.
	
	A JSON object is encoded into the note text for IONOTE and NONOTE note types.
	
	{
	"action":"string",
	"milestone":"string",
	"delayReason":"string",
	"rescheduleFrom":"YYYY-MM-DD"
	}
	
	Fields that are not in use, will be excluded and should be concidered null.
	
	Possible IONOTE's:
	
	"{"action":"milestone","milestone":"newInfrastructureStatus_0"}"
	
	"{"action":"milestone","milestone":"newInfrastructureStatus_10"}"
	
	"{"action":"milestone","milestone":"newInfrastructureStatus_20"}"
	
	"{"action":"milestone","milestone":"newInfrastructureStatus_30"}"
	
	"{"action":"delay","delayReason":"customer","rescheduleFrom":"2018-05-12"}"

	"{"action":"delay","delayReason":"IO","rescheduleFrom":"2018-05-12"}" "{"action":"delayedCompletion","completionETA":"2019-01-25"}" Possible NONOTE's: "{"action":"milestone","milestone":"provisioningDataReadyForSP"}" "{"action":"delay","delayReason":"NO","rescheduleFrom":"2018-05-12"}" "{"action":"delayedCompletion","completionETA":"2019-01-25"}" Sample GENOTE's (plain text): "Delayed due to frozen ground" (In case of delay or delayedCompletion, a GENOTE must be added, contain a meningfull description, to let SP know why work has been delayed.) The note text may not exceed 80 charactors.
Response (http response code 200)	**Description**
null	Note added successfully
Response (http response code 400)	**Description**
responseCode	"FAILURE"
message	"what went wrong"
Response (http response code 500)	**Description**
null	Note not added due to internal server error

Table 15: Data definition

Description	**JSON**

Request	{
	"serviceSubscriptionID":"100000079",
	"id":8001,
	"noteType":"IONOTE"
	"note":["{\"action\":\"milestone\",\"milestone\":\"newInfrastructureStatus_10\"}"]
	}

Response	Null

Table 16: Data definition

10.3 Events

The event service is used to create events, that will be used to notify the Service Provider through the order fulfillment process. Events can also be used to communicate between NO and IO entities during the lifetime of a service, and although the createEvent, getNextEvent and updateEvent API's are defined under this Order Fulfilment section, NO's and IO's can use the feature to communicate outside of the order fulfilment process if needed.

The service will have the following operations:

- createEvent
- getNextEvent

- updateEvent

10.3.1 Create Event

The sequence for create event is showed below:

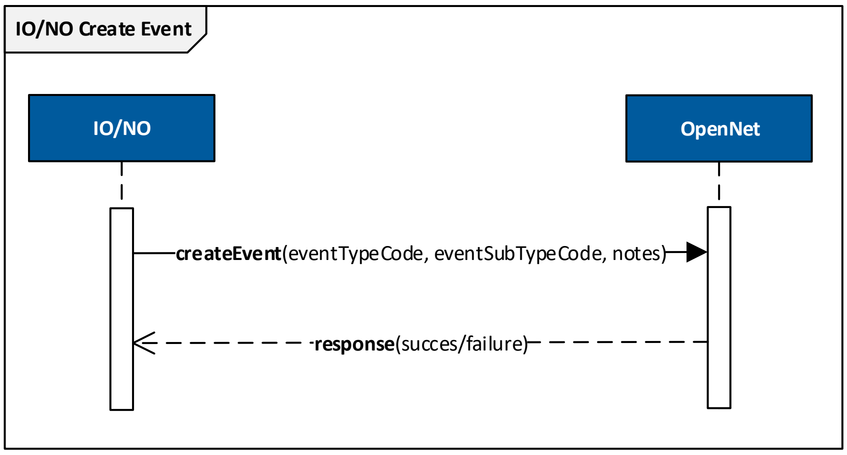


Figure 13: Events

10.3.1.1 Data definition

Input - Data Item	Description
accountNo	Service Provider account number. Disregard in this context.
serviceSubscriptionID	Unique identifier of the main service for which capacity is reserved. This unique identifier is constant across order fulfillment process and stays the same, even during a possible option change before the original order has been fulfilled.
eventTypeCode	Type of the event: "ODRFUL" = Order fulfillment "NOEVENT" = Event passed from NO->IO "IOEVENT" = Event passed from IO->NO Note that the NOEVENT and IOEVENT types are generic event, and the content of the event note must be defined for the specific NO-IO relationship. The content will not be validated by OpenNet.
eventSubTypeCode	Subtype of the event [ODRFUL]: "STACH" = statusChanged "ODRCOM" = completed "PROVSP" = provisioningDataReadyForSP "REQRES" = requestRescheduling "REQFRE" = requestFreeResheduling "GENUPD" = generalUpdate Subtype of the event [NOEVENT/IOEVENT]: "TYPE1" = Generic event sub type. Usage to be defined by NO/IO TYPE2" = Generic event sub type. Usage to be defined by NO/IO "TYPE3" = Generic event sub type. Usage to be defined by NO/IO "TYPE4" = Generic event sub type. Usage to be defined by NO/IO "TYPE5" = Generic event sub type. Usage to be defined by NO/IO
Notes	For orderFulfillment eventTypes of specific subTypes, the notes field will contain a JSON encoded string:

	A JSON object is encoded into the note text. See below for details of the notes field content. For NOEVENT and IOEVENT the content of the notes will be defined by the NO/IO entities.
assignee->type	For serviceSubscriptionID related events, always use "ENTITY"
assignee->value	For orderFulfillmentEvents always assign to "SP" For NOEVENT always assign to "IO" For IOEVENT always assign to "NO"
Response (http response code 200)	**Description**
event data	Complete event with data is returned.
Response (http response code 400)	**Description**
responseCode	"FAILURE"
Message	"what went wrong"
Response (http response code 500)	**Description**
NULL	Event creation could not be completed due to internal server error

Table 15: Data Definition

Description	**JSON**

Request	Create ODRFUL event:
	{
	  "serviceSubscriptionID":"100000079",
	  "eventTypeCode":"ODRFUL",
	  "eventSubTypeCode":"STACH",
	  "notes":["{\"basketID\":2005,\"action\":\"milestone\",\"milestone\":\"newInfrastructureStatus_10\"}"],
	  "assignee":{
	    "type":"ENTITY",
	    "value":"SP"
	  }
	}
	
	Create NOEVENT event:
	{
	  "serviceSubscriptionID":"100000079",
	  "eventTypeCode":"NOEVENT",
	  "eventSubTypeCode":"TYPE1",
	  "notes":["DATA TO BE DEFINED BY NO/IO"],
	  "assignee":{
	    "type":"ENTITY",
	    "value":"IO"
	  }
	}
	
	Create IOEVENT event:
	{
	  "serviceSubscriptionID":"100000079",
	  "eventTypeCode":"IOEVENT",
	  "eventSubTypeCode":"TYPE1",
	  "notes":["DATA TO BE DEFINED BY NO/IO"],
	  "assignee":{
	    "type":"ENTITY",
	    "value":"NO"

	} }

Response	{
	  "id": 5262
	}

Table 17: Data Definition JSON format

Event Type	**Event Sub Type**	**Releated to a service Subscription ID**	**Usage**	**Format of note text**

orderFulfillment	statusChanged	Yes	Created by IO when updates to an order is created.
			
			This event subtype is used while processing an WORK_ORDER_IO fulfillment request.
				A JSON object is encoded into the note text.
				
				{
				"basketID":integer,
				"action":"string",
				"milestone":"string",
				}

orderFulfillment	Request Rescheduling	Yes	Created by NO/IO in case the delivery is delayed, due to the end customer.
			
			This event subtype is used while processing any fulfillment request where an agreed date or appointment is expected to for order completion.
				A JSON object is encoded into the note text.
				
				{
				"basketID":integer,
				"action":"string",
				"delayReason":"string",
				"rescheduleFrom":"YYYY-MM-DD"
				}

orderFulfillment	requestFree Rescheduling	Yes	Created by NO/IO in case the delivery is delayed, due to NO or IO.
			
			This event subtype is used while processing any fulfillment request where an agreed date or appointment is expected to for order completion.
			
			OpenNet will automatically waive the later rescheduling fee, othervise charged to the SP.
				A JSON object is encoded into the note text.
				
				{
				"basketID":integer,
				"action":"string",
				"delayReason":"string",
				"rescheduleFrom":"YYYY-MM-DD"
				}

orderFulfillment	provisioningDataReadyForSP	Yes	Created by NO when provisioning data is ready for SP as part of the getFulfillmentStatus call.
			
			This event subtype is used exclusively when processing the WORK_ORDER_NO fulfillment request.
				A JSON object is encoded into the note text.
				
				{
				"basketID":integer,
				"action":"string",
				"milestone":"string",
				}

orderFulfillment	completed	Yes	Raised by the OpenNet system, when fulfillment is completed for an order and the recurrent billing starts.	Empty

			No JSON is included here, link to order is via serviceSubscriptionID.	

orderFulfillment	generalUpdate	Yes	Raised by NO/IO to signal order completion is delayed, but under NO/IO control. SP do not need to contact end customer or reschedule, as communication with the customer is already handled.
			
			A descriptive reason for the delay, will be provided as a GENOTE which can be read in getFulfillmentStatus API.
				A JSON object is encoded into the note text.
				{
				"basketID":integer,
				"action":"string",
				"completionETA":"YYYY-MM-DD"
				}

Table 18: Event Type & Format

Event type	Event Sub Type	Example
orderFulfillment	statusChanged	"{\"basketID\":2004, \"action\":\"milestone\",\"milestone\":\"newInfrastructureStatus_0\"}" "{\"basketID\":2004, \"action\":\"milestone\",\"milestone\":\"newInfrastructureStatus_10\"}" "{\"basketID\":2004, \"action\":\"milestone\",\"milestone\":\"newInfrastructureStatus_20\"}" "{\"basketID\":2004, \"action\":\"milestone\",\"milestone\":\"newInfrastructureStatus_30\"}"
orderFulfillment	requestRescheduling	"{\"basketID\":2004, \"action\":\"delay\",\"delayReason\":\"customer\",\"rescheduleFrom\":\"2018-05-12\"}"
orderFulfillment	requestFreeRescheduling	"{\"basketID\":2004, \"action\":\"delay\",\"delayReason\":\"IO\",\"rescheduleFrom\":\"2018-05-12\"}" "{\"basketID\":2004, \"action\":\"delay\",\"delayReason\":\"NO\",\"rescheduleFrom\":\"2018-05-12\"}"
orderFulfillment	provisioningDataReadyForSP	"{\"basketID\":2004, \"action\":\"milestone\",\"milestone\":\"provisioningDataReadyForSP\"}"
orderFulfillment	generalUpdate	"{\"basketID\":2004, \"action\":\"delayedCompletion\",\"completionETA\":\"2019-01-25\"}"

Table 19: Event Notes

10.3.2 Get Next Event

The Get Next Event service enables the Network Operator to get IOEVENT's and Infrastructure Owner to get NOEVENT's.

The sequence diagram for Get Next Event is shown in the figure below**:**

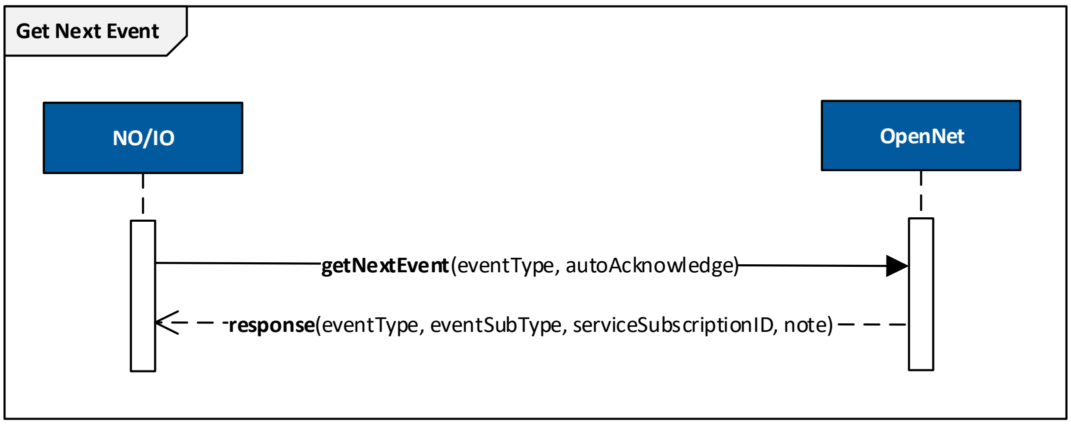

Get Next Event
NO/IO
OpenNet
getNextEvent(eventType, autoAcknowledge)
response(eventType, eventSubType, serviceSubscriptionID, note)

Figure 14: Get Next Event

10.3.2.1 Data Definition

The following provides a high-level view of the data items expected to support the INPUT and RESPONSE payload. This data is will be subject to change as design/development evolves through.

Operation: getNextEvent

Input - Data Item	**Required**	**Description**h1
eventType [query param]	Yes	Type of event: IO's should specify "NOEVENT" NO's should specify "IOEVENT"
autoAcknowledge [query param]	No	If false then same event will be returned by this request until actively acknowledge by the SP. Using the autoAcknowledge=false feature, enables SP's to ensure event handling finished successfully before event is removed from the OpenNet queue. If not included the autoAcknowledge value defaults to true.
Response (http response 200 – OK)		
id	Id of the event, e.g. 6688	
eventTypeCode	Type of the event: "NOEVENT" or "IOEVENT"	
eventSubTypeCode	Subtype of the event: "TYPE1" = Generic event sub type. Usage to be defined by NO/IO TYPE2" = Generic event sub type. Usage to be defined by NO/IO "TYPE3" = Generic event sub type. Usage to be defined by NO/IO "TYPE4" = Generic event sub type. Usage to be defined by NO/IO "TYPE5" = Generic event sub type. Usage to be defined by NO/IO	
accountNo	Account not related to the event	
serviceSubscriptionID	The serviceSubscriptionID to which this event relates.	
creationDate	Date and time when event was created	
note	Free text with date and author of the note. Maximum length of text is 2000 characters	
assignee	Type and value of the assignee	
Response (http response 204 – No Content)		
No events available		

Table 20: Data items - getNextEvent

10.3.3 *Update Event*

This service enables the Network Operator and Infrastructure Owner to acknowledge/close an event received earlier. The use of this service is only relevant if the NO's and IO's also use the autoAcknowledge=false feature in the Get Next Event API.

The sequence diagram for Get Fulfillment Status is shown in the figure below:

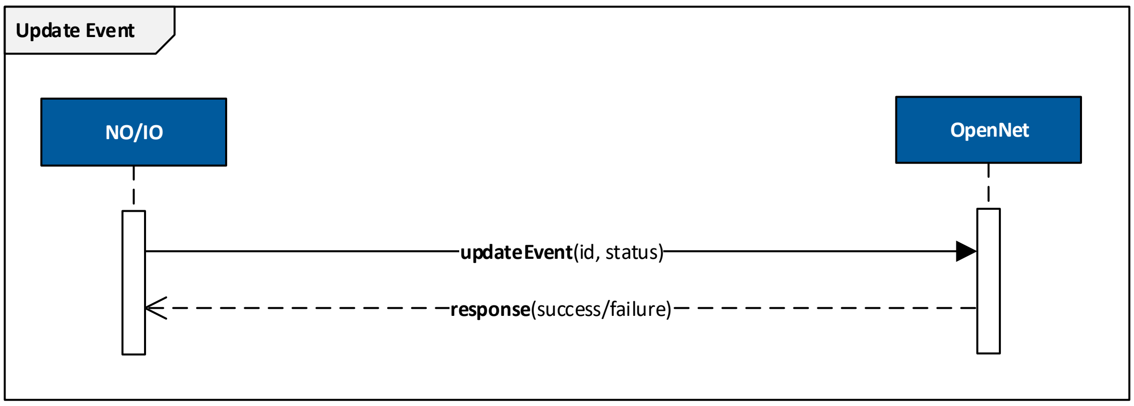


Figure 15: Update Event

10.3.3.1 Data Definition

The following provides a high-level view of the data items expected to support the INPUT and RESPONSE payload

Operation: Update event

Input - Data Item	**Required**	**Description**
id	Yes	ID of the event to update.
status	Yes	New status of event. Use the value "CLOSED" to acknowledge and remove event from the event queue. Use the value "OPEN" to reopen an earlier closed event to reinstate it it the event queue.
Response (http response 204 – No content)		
The event was updated successfully		
Response (http response 400 – Bad request)		
responseCode	Error classification: "FAILURE"	
message	Description of the failure reason	
Response (http response 500 – Internal server error)		
Event was not updated due to an internal server error.		

Table 21: Data items – Update Event

10.4 UDF API

This API is used to update the UDF's within Cerillion Core. The UDF's are used to parse data between NO and IO, and to provide provisioning data from NO to SP. The sequence for Update UDF is showed below:

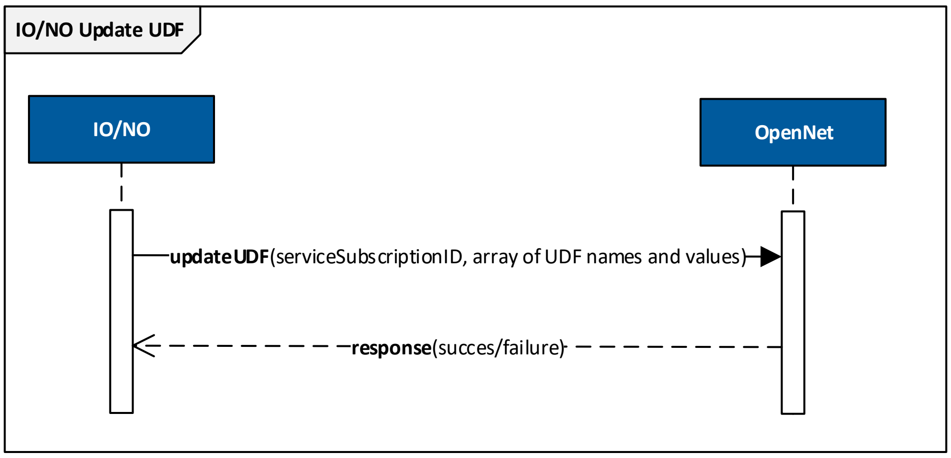

IO/NO Update UDF
IO/NO
OpenNet
updateUDF(serviceSubscriptionID, array of UDF names and values)
response(succes/failure)

Figure 16: Update UDF

10.4.1 Post UDF

Input - Data Item	**Description**
serviceSubscriptionID	Identifier of the service for which the UDF(s) are to be set.
udfDetails	An array of key value pairs, where udfName is the key, and udfValue is the content. NO/IO can manipulate the 18 UDF fields named: networkOperatorOut1..networkOperatorOut6, infrastructureOwnerOut1..infrastructureOwnerOut6, ServiceProviderOut1..ServiceProviderOut6
Response(http response code 204)	Description
NULL	UDF(s) updated successfully
Response(http response code 400)	Description
responseCode	"FAILURE"
message	"what went wrong"
Response (http response code 500)	Description
NULL	UDF(s) not updated due to internal server error

Table 22: Data definition

10.5 Combined Use of UDF's, Notes, Events & Fulfillment Requests

The overall usage of notes, events and fulfillment requests, ensures that NO/IO receives correct information about what is expected of the entities. More over the usage of the three API's ensure that the order history is captured using order notes, while at the same time providing the most important data directly to the Service Provider via Events.

Please note that the use of updateUDF, addNote and createEvent is only expected as part of WORK_ORDER_NO and WORK_ORDER_IO. For the other Fulfillment Request, OpenNet only expect NO/IO to complete the requests when the tasks are done.

There are a few rules that must be applied before the entities using the OpenNet platform can rely on the data flowing from SP to NO/IO and back again from NO/IO to SP.

1. History must be updated before creating events. If SP reacts to an event, by collecting the complete fulfillment status of an order, the history must be available.
2. serviceProviderOut UDF's must be updated before adding notes or creating events about the milestone provisioningDataReadyForSP.
3. Before creating the orderFulfillmentEvent with subType provisioningDataReadyForSP, all necesary serviceProviderOut UDF fields must be updated.
4. Before completing a Fulfillment Request, all relevant notes, UDF and events must have been added/updated/created.
5. In case of a delay that required rescheduling to be done by the Service Provider, NO/IO are required to first add an NONOTE/IONOTE with structured data. Then a GENOTE must be added with more detailed information as to why delivery has been delayed and lastly a requestRescheduling/requestFreeRescheduling event must be created.
6. If a delay is caused by the end-customer, the eventSubType "requestRescheduling" must be used.
7. If a delay is caused by NO/IO, the eventSubType "requestFreeRescheduling" must be used, as this will make OpenNet waive the normal rescheduling fee for the later rescheduling.

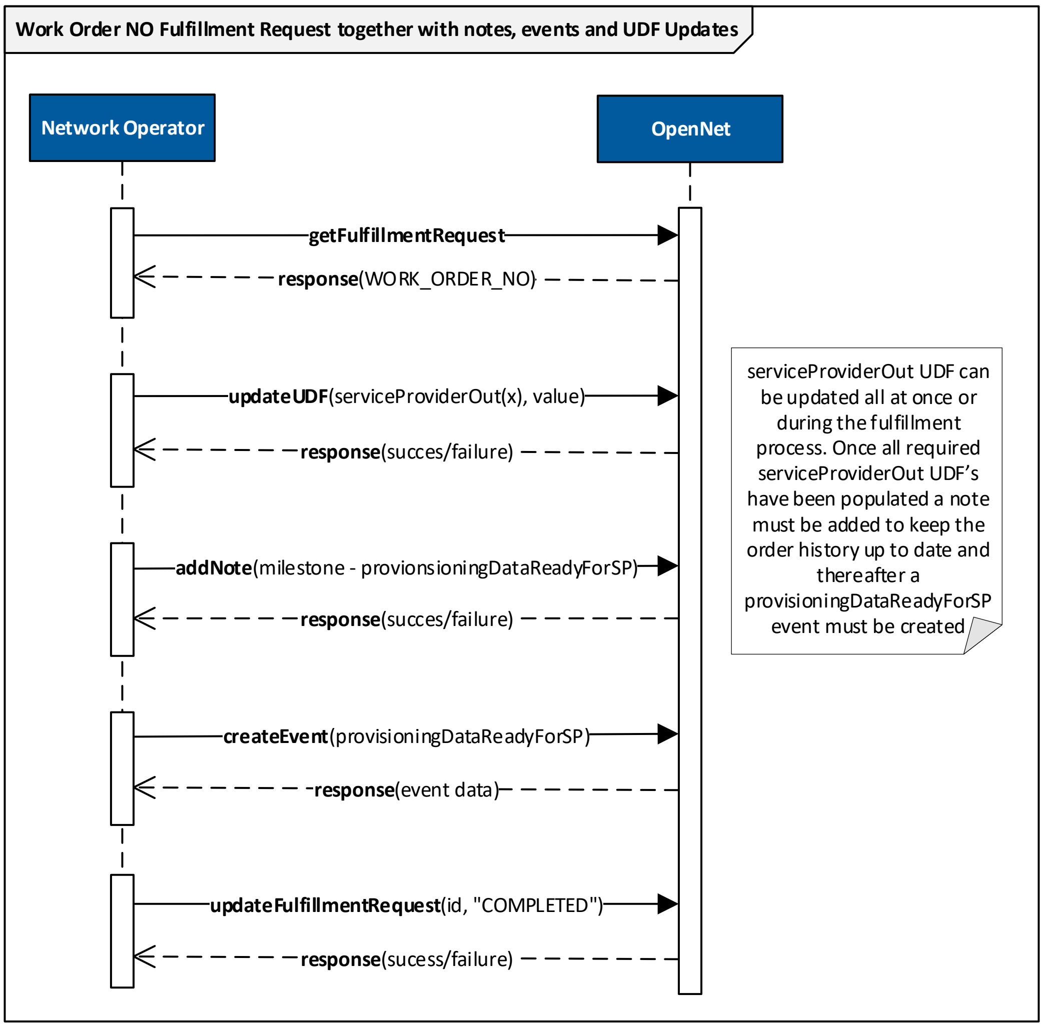

Work Order NO Fulfillment Request together with notes, events and UDF Updates
Network Operator
OpenNet
getFulfillmentRequest
response(WORK_ORDER_NO)
updateUDF(serviceProviderOut(x), value)
response(succes/failure)
addNote(milestone - provionsioningDataReadyForSP)
response(succes/failure)
createEvent(provisioningDataReadyForSP)
response(event data)
updateFulfillmentRequest(id, "COMPLETED")
response(sucess/failure)
serviceProviderOut UDF can be updated all at once or during the fulfillment process. Once all required serviceProviderOut UDF's have been populated a note must be added to keep the order history up to date and thereafter a provisioningDataReadyForSP event must be created

Figure 17: WORK_ORDER_NO Fulfillment request

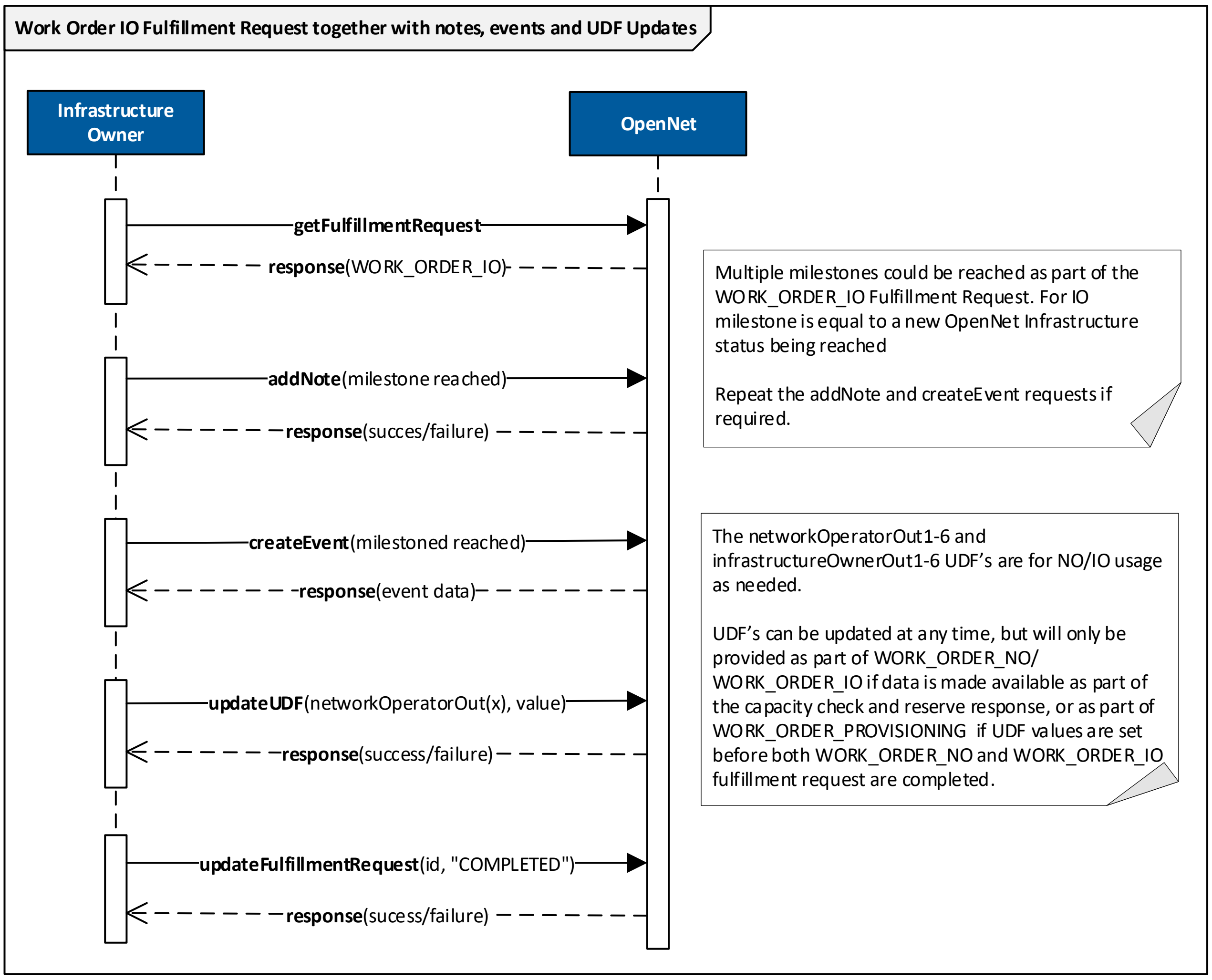

Work Order IO Fulfillment Request together with notes, events and UDF Updates
Infrastructure Owner
OpenNet
getFulfillmentRequest
response(WORK_ORDER_IO)
addNote(milestone reached)
response(succes/failure)
createEvent(milestoned reached)
response(event data)
updateUDF(networkOperatorOut(x), value)
response(success/failure)
updateFulfillmentRequest(id, "COMPLETED")
response(sucess/failure)
Multiple milestones could be reached as part of the WORK_ORDER_IO Fulfillment Request. For IO milestone is equal to a new OpenNet Infrastructure status being reached
Repeat the addNote and createEvent requests if required.
The networkOperatorOut1-6 and infrastructureOwnerOut1-6 UDF's are for NO/IO usage as needed.
UDF's can be updated at any time, but will only be provided as part of WORK_ORDER_NO/ WORK_ORDER_IO if data is made available as part of the capacity check and reserve response, or as part of WORK_ORDER_PROVISIONING if UDF values are set before both WORK_ORDER_NO and WORK_ORDER_IO fulfillment request are completed.

Figure 18: WORK_ORDER_IO Fulfillment Request

11. Service Assurance

11.1 Support Tools (OpenNet<->NO/IO)

Support Tools allow the Service Provider to run diagnostics on a service and perform specific update operations. The request will send and retrieve data directly from the Network Operator and, when requested, perform an update to the service.

The sequence diagram for Support Tools is showed in the figure below:

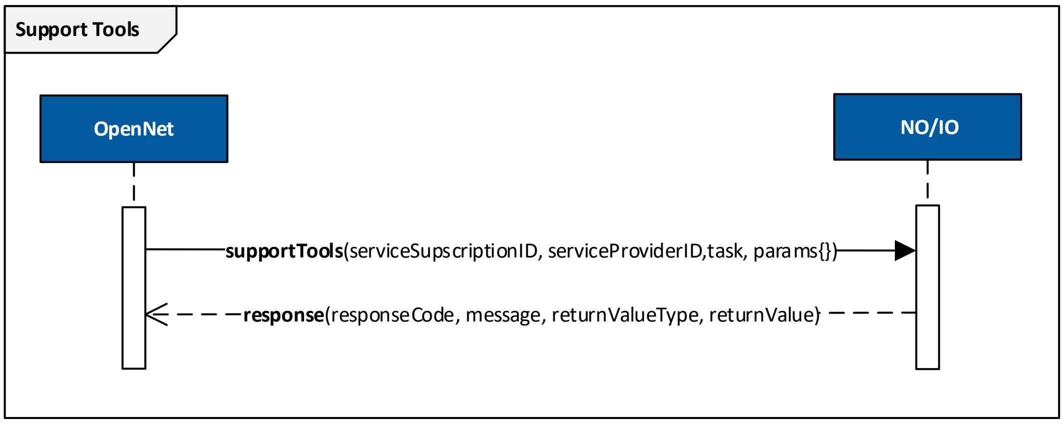

Support Tools
OpenNet
NO/IO
supportTools(serviceSupscriptionID, serviceProviderID,task, params{})
response(responseCode, message, returnValueType, returnValue)

Figure 19: Support Tools

11.1.1 General Data Definition

The following provides a high-level view of the data items expected to support the INPUT and RESPONSE payload. Support Tools is build to support further tasks going forward, but for now the only task possible is to get system data about an active service.

Input - Data Item	**Description**
serviceSubcriptionID	Unique service identifier
task	String – e.g. ("getServiceStatus")
serviceProviderID	String
params{}	Defaut null, Object optional
Response	
responseCode	Can be 0 or 1. 0: Response is valid 1: Error happened. See message for details.
message	String which hold error information if responseCode is 1
returnValueType	String defining which structure is returned in returnValue object.
returnValue	Null or Object.

Table 23: Data definition

11.1.2 Detailed Data Definition for task "getServiceStatus" version 1.0

Input - Data Item	**Description**	**Main Product Relevance**
serviceSubcriptionID	The unique service identifier	
task	"getServiceStatus"	
serviceProviderID	String	
params{}	null	
Response		
responseCode	Can be 0 or 1. 0: Response is valid 1: Error happened. See message for details.	All
message	String which hold error information if responseCode is 1	All
returnValueType	"fibreServiceStatus"	All
returnValue	Object returned to SP holding data from NO systems.	All
returnValue.configuration		All

returnValue.configuration.qosProfile	Possible values: QOSPROFILE1, QOSPROFILE2, QOSPROFILE3. "Null" or "" if no qosProfile is configured.	H1,H2
returnValue.configuration.multicastSnoop	Array of snooped IP addresses	H1
returnValue.configuration.provisionedVlans	Array of C-VLAN tags provisioned for the specified service.	H1
returnValue.configuration.provisionedVlans[index].id	C-VLAN Id as a string. If none are configured then id is returned as an empty array or null	H1
returnValue.configuration.provisionedVlans[index].tagged	Boolean stating if a VLAN is tagged or untagged.	H1
returnValue.configuration.provisionedVlans[index].multicast	Boolean stating if multicast is enabled for the VLAN.	H1
returnValue.configuration.adminState	Possible values: LOCKED, UNLOCKED The administrative state (admin state) is a parameter describing whether the physical port is enabled or disabled by configuration. This is in most cases set by provisioning, but could be in other scenarios be done manually by a network operator, by the software in some CPE's if a lot of errors are seen on the interface, or lastly if the port by default is disabled in factory settings and the provisioning haven't enabled it.	All
returnValue.configuration.rfOutputSignalStrength	Possible values: HIGH, MEDIUM, LOW This property will be null if service is not H3.	H3
returnValue.configuration.rfFilter	Possible values: LARGE, MEDIUM, SMALL This property will be null if service is not H3.	H3
returnValue.configuration.port	For H1 services: ethPort0, ethPort1, ethPort2, ethPort3, ethPort4, ethPortn (n being a number, depending on what is printed on the CPE port) For H3 services: rfPort (no number for this port. Only one exist.) Unless a port change has been triggered, this string will match the string parsed back during original sale.	H1,H3
returnValue.configuration.maxNumberOfMACAddresses	A number. Posibilities depends on product. 8, 64, 128, 256, 1024, 2048, 4096	H1,H2
returnValue.accessConnection.accessTechnology	Possible values at this time: GPON, PTP	All
returnValue.accessConnection.linkState	Possible values: UP, DOWN If DOWN is returned, NO is unable to detect the CPE equipment.	All
returnValue.accessConnection.opticalSignalPower	Optical Signal Power messured at the POP	All
returnValue.accessConnection.opticalSignalPower.rx	CPE->POP	All
returnValue.accessConnection.opticalSignalPower.rx.waveLength	Wavelength used	All
returnValue.accessConnection.opticalSignalPower.rx.value	Measured value, "null" is returned if the port is down	All
returnValue.accessConnection.opticalSignalPower.rx.inRange	NO has the responsibility to state if the measured value is in range and return true or false in this Boolean.	All
returnValue.accessConnection.opticalSignalPower.tx	POP->CPE	All

returnValue.accessConnection.opticalSignalPower.tx.waveLength	Wavelength used	All
returnValue.accessConnection.opticalSignalPower.tx.value	Measured value, "null" is returned if the port is down	All
returnValue.accessConnection.opticalSignalPower.tx.inRange	NO has the responsibility to state if the measured value is in range and return true or false in this Boolean.	All
returnValue.accessConnection.lineErrors	Line errors counted in POP. NO will not reset this counter at fixed intervals, and SP should call support tools API with an interval to look at change to detect errors during error investigation.	All
returnValue.accessConnection.lineErrors.errorIn	Errors counted inbound (CPE->POP)	All
returnValue.accessConnection.lineErrors.errorOut	Errors counted outbound (POP->CPE)	All
returnValue.terminationPoint		H1,H3
returnValue.terminationPoint.cpeType	Name of CPE type. New types can be introduced. Ex: Nokia G-041	H1,H3
returnValue.terminationPointcpeStatus	Possible values: UP, DOWN Resons for returning DOWN includes but is not restricted to CPE currently restarting or CPE not being able to boot due to an error.	H1,H3
returnValue.terminationPoint.ethernet		H1
returnValue.terminationPoint.ethernet.status	Possible values: UP, DOWN Status will be UP if the port is connected to any workable Ethernet port and a link can be brought up.	H1
returnValue.terminationPoint.ethernet.learnedMACAddresses	Array of learned MAC addresses	H1
returnValue.terminationPoint.ethernet.bitrate	Currently possible values are: FE = Fast Ethernet GE = Gigabit Ethernet (Also for speeds above 1Gbps as version 1 is deprecated)	H1
returnValue.terminationPoint.ethernet.duplex	Possible values: HDX, FDX	H1
returnValue.terminationPoint.ethernet.dhcp	Not used – Values should be disregarded	
returnValue.terminationPoint.ethernet.dhcp.rx	Not used – Values should be disregarded	
returnValue.terminationPoint.ethernet.dhcp.tx	Not used – Values should be disregarded	

Table 24: Data items - Support tools "getServiceStatus" version 1.0

11.1.3 Detailed Data Definition for task "getServiceStatus_v2"

The following will describe each property in the response and possible values. A general rule exists, that if a property is not available it can be either null or omitted. For array properties null, omitted or empty array [] are allowed.

StatusCode is to be set on every element as described in schema and example with the following list of codes:

Code	**Description**
1	Property not available on current technology
2	Property is configured data read from configuration
3	Property is operational live data read from network device

Table 25: : Data items – Property status codes for task getServiceStatus_v2

Input - Data Item	**Description**
serviceSubcriptionID	The unique service identifier
Task	"getServiceStatus_v2"
serviceProviderID	String
params{}	null

Table 26: : Data items - Input parameters for task getServiceStatus_v2

Response				
Property Name	**Type**	**Required**	**Relevant to**	**Description**
responseCode	int	Yes	H1, H2, H3	0: Success 1: Error
message	string	Yes	H1, H2, H3	If responseCode=1 then this string must hold a meaningful text to enable SP to know what went wrong.
returnValueType	string	Yes	H1, H2, H3	fibreServiceStatus_v2
returnValue.	object			
configuration.	object			
adminLocked	boolean		H1, H2, H3	Parameter describing if the service has been administratively locked where the IO prefers to do so, and therefore would be down by actively doing so. e.g., if the service is barred. false: UNLOCKED true: LOCKED
maxNumberOfMACAddresses	int		H1, H2	Possibilities depends on product. e.g., 8, 64, 128, 256, 1024, 2048, 4096
port	string		H1, H3	e.g., ethPort0, ethPort1, ethPort2, ethPort3, ethPort4 etc. ethPortn (n being a number, depending on what is printed on the CPE port) rfPort (no number for this port. Only one exist.) Unless a port change has been triggered, this string will match the string parsed back during original sale.
qosProfile	string		H1, H2	Name of QoS profile: e.g., QOSPROFILE1, QOSPROFILE2, QOSPROFILE3
accessConnection.	object			
accessTechnology	string		H1, H2, H3	GPON PTP
linkUp	boolean		H1, H2, H3	Parameter describing if the connection is seen as up from

				the access node towards the CPE (H1) or termination point (H2) false: DOWN true: UP
optical.	object			
upStream.	object			
wavelength	int		H1, H2	Wavelength used from CPE or termination point towards POP.
signalPower	int		H1, H2	Signal power measured from CPE or termination point towards POP.
inRange	boolean		H1, H2	Network Operator is responsible for signaling if the above reported signalPower is within the acceptable range.
downStream.	object			
wavelength	int		H1, H2	Wavelength used from POP towards CPE or termination point.
signalPower	int		H1, H2	Signal power measured from POP towards CPE or termination point.
inRange	boolean		H1, H2	Network Operator is responsible for signaling if the above reported signalPower is within the acceptable range.
lineError.	object			
upStream	int		H1, H2	Errors counted from CPE or termination point towards POP
downStream	int		H1, H2	Errors counted from POP towards CPE or termination point.
terminationPoint.	object			
cpeType	string		H1, H3	Name of CPE type. Available types are presented by IO in "Fiber Boks" document on OpenNet partner portal. New types can be introduced.
serial	string		H1, H3	Serial number of the CPE.
cpeUp	boolean		H1, H3	Parameter describing if the CPE is seen as up and online false: DOWN true: UP
ethernet.	object			
linkUp	boolean		H1	Status of the link on the ethernet port towards the SP or costumer equipment. false: DOWN true: UP
bitrate	int		H1	Bitrate of the ethernet port towards the SP or costumer equipment. Bitrate value measured in Mbps e.g., 10, 100, 1000, 2500

				Value should be null if link is down
fullDuplex	boolean		H1	Duplex of the ethernet port towards the SP or costumer equipment. False: Not full duplex true: Full duplex Value should be null if link is down
rf.	object			
rfOutputSignalStrength	string		H3	The amount of signal strength output (dB) on the RF port. HIGH MEDIUM LOW AUTO
rfFilter	string		H3	The frequency package passed through the configurable RF filter on the RF port. LARGE MEDIUM SMALL
optical.	object			
downStream.	object			
Wavelength	int		H3	Wavelength used for CATV from POP towards CPE or termination point.
signalPower	int		H3	Signal power on above wavelength measured from POP towards CPE or termination point.
inRange	boolean		H3	Network Operator is responsible for signaling if the above reported signalPower is within the acceptable range.
vLans.	array			
cVlan	int		H1	Configured C-VLAN
tVlan	int		H1	Configured T-VLAN
Tagged	boolean		H1	Whether or not the current C-VLAN in the array is tagged false: Untagged true: Tagged
Multicast	boolean		H1	Whether or not the current C-VLAN in the array has multicast enabled. false: Multicast Disabled true: Multicast Enabled
.multicastSnoop.	array			

	string		H1	IGMP snooped address seen on the service wherever the IO can do so.
.learnedMACAddresses.	array			
Mac	string		H1, H2	Learned MAC address in standard IETF/IEEE format
onPOI	boolean		H1, H2	false: MAC address is learned on the termination point true: MAC address is learned on the SP facing POI interface
poi.	object			
id.value	string		H1, H2	Identification known by the SP of the POI.
sVlan.value	int		H1, H2	Configured S-VLAN

Table 27: Data items - Response for task getServiceStatus_v2

11.2 Support Tools (NO/IO<->OpenNet<->NO/IO)

In scenarios where IO and NO are two separate entities, it is possible for an IO to call the Support Tools API (Same as SP) for services related the the calling IO. By doing this the IO can gain knowledge about active services where required. One example of such a scenario is when an IO technician are at the customer site to correct an issue.

The sequence diagram for Support Tools is showed in the figure below:

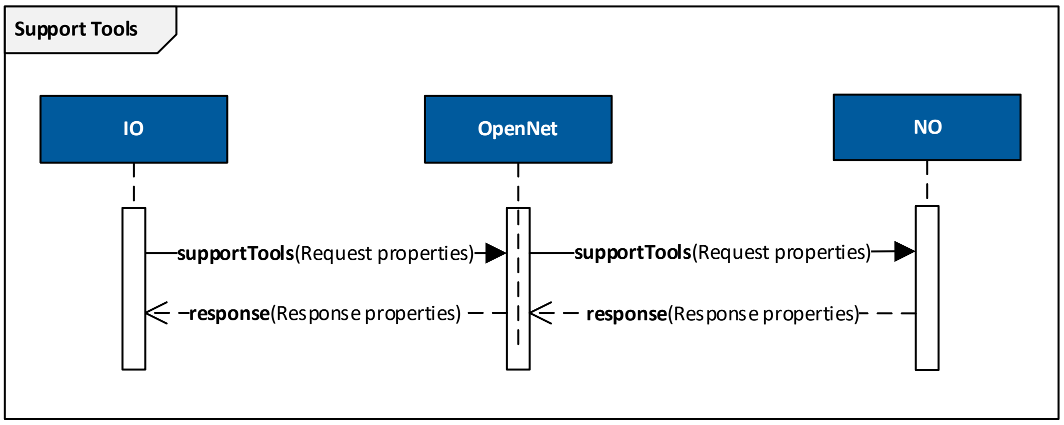

Support Tools
IO
OpenNet
NO
supportTools(Request properties)
response(Response properties)
supportTools(Request properties)
response(Response properties)

Figure 20: Support Tools

11.2.1 General Data Definition

The following provides a high-level view of the data items expected to support the INPUT and RESPONSE payload.

For more detailed description of each property see section: 11.1 Support Tools (OpenNet<->NO/IO)

Input - Data Item	**Description**
serviceSubcriptionID	Unique service identifier
task	String

	Possible values are: getServiceStatus getServiceStatus_v2
params{}	Defaut null, Object optional
Response	
responseCode	Can be 0 or 1. 0: Response is valid 1: Error happened. See message for details.
message	String which hold error information if responseCode is 1
returnValueType	String defining which structure is returned in returnValue object.
returnValue	Null or Object.

Table 28: Data definition

11.3 Trouble Ticket

The OpenNet system provides an ability for SP's to raise, update and get trouble tickets on behalf of their end-customer in the case of incidents relating to a service. NO's and IO's are able to get and update trouble tickets.

Before a Service Provider creates a trouble ticket, the Service Provider is required to perform 1st line triage related activities to validate the end-customer issue. This includes but is not restricted to calling the OpenNet supportTools API, to get the current status of the serviceSubscriptionId in question.

Services will be provided for the following:

- getTroubleTicket
- updateTroubleTicket
- getAllTroubleTicketsAssignedToUser
- searchForOpenTroubleTickets

11.3.1 Get Trouble Ticket

The Get Trouble Ticket service allows the NO/IO to retrieve the content of an incident or service request by using the troubleTicketId.

The sequence diagram for getTroubleTicket is shown in the figure below**:**

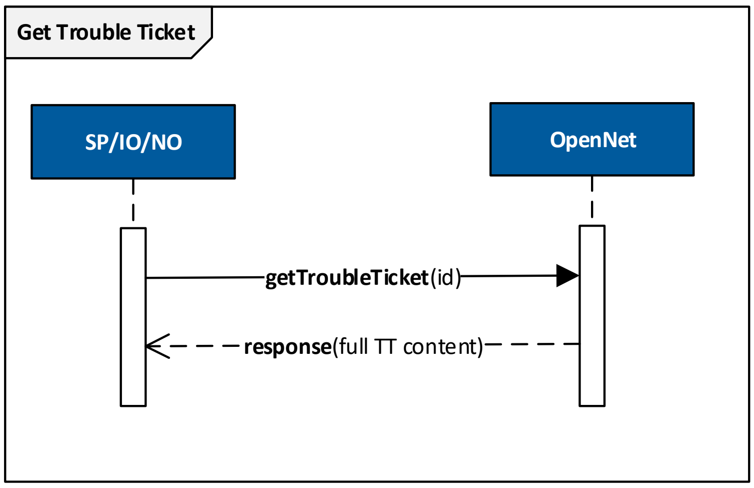


Figure 21: Get Trouble Ticket

11.3.1.1 Data Definition

The following provides a high-level view of the data items expected to support the INPUT and RESPONSE payload.

Operation: getTroubleTicket

Input - Data Item	**Description**
Id	Ticket ID being fetched
Response	
classification	For STAN tickets the following classification codes are used: Classification code can be "MAJ" = Major "MIN" = Minor "IMI" = Immediate Service (Only allowed for H1 and H3 services) "SER" = Service Request (Only account level trouble tickets. Not in use currently.) "OTH" = Other (Not in use) For serviceSubscriptionID related incidents use "MAJ", "MIN". For H1/H3 services "IMI" is also possible. For SANA and SAWA tickets, the possible classifications are contract bound. Sample values are: “BRO”: Broadband “WIF”: WiFi “TVS”: TV/STB “VOI”: VOIP

	For Network tickets the possible classifications are: “POI” = POI related issues “MUL” = Issue identified to impact multiple services
outageId	If NO/IO link this Trouble Ticket to an outage, they can provide the link in this field.
externalIdSP	Exclusivly used by SP
externalIdNO	Exclusivly used by NO
externalIdIO	Exclusivly used by IO
description	Free text field to enter high-level description of the issue, i.e. a heading.
Severity	For “STAN” tickets this property must not be defined by SP’s, as it is set by OpenNet based on the SLA bought together with the service identified by the Service Subscription ID. STANDARD SLA: 50 BRONZE SLA: 60 SILVER SLA: 70 GOLD SLA: 80 For “SANA” and “SAWA” tickets severity is used to specify the SLA the IO are to work by, and thereby also the cost of the work ordered by the SP. Possible values are contract bound. For “NETW” tickets severity is always set to 99.
Type	Type of Trouble Ticket. "STAN" = Standardard ticket to handle errors with infrastructure equipment or config “SANA” = SP Service Assurance (no appointment) used to support errors on SP equipment in scenarios where contract states that IO technician must arrive on site very quickly, or IO is the entity agreeing a timeslot with the end customer. “SAWA” = SP Service Assurance (with appointment) used to support errors on SP equipment where contracts states it is the SP who schedules the appointment with the end customer. “NETW” = Network ticket to handle POI related issues or network issues where a SP has identified a larger number of impacted services. NOTE: Not all IO’s support SANA and SAWA ticket types, and it requires contractural agreement to agree scope and afterwards use the functionality.
serviceSubscriptionId	Service subscription id to which this Trouble Ticket belong
accountNo	This identifies the related Service Provider if a Service Subscription ID is defined. If this is a “NETW” tickets where no Service Subscription ID is defined, this field will hold the specific account number for the SP<->IO relation. See “relatedEntities” property for the easiest way to identify the specific entities related to a ticket regardless of it being linked to a specific Service Subscription ID or not.
contactName	Contact name for reference
contactNumber	Contact number for reference
activityTask	Not used

Assignee	Record
assignee->type	When trouble ticket is assigned to SP, assignee type will be user in the response. When trouble ticket is assigned to NO/IO, assignee type will be department in the response.
assignee->value	
Id	The Id of this Trouble Ticket
Status	Main status of this trouble ticket. OPEN or CLOSED.
creationDate	When was this trouble ticket created
Note	Array of records
note[index]->date	Datetime stamp for when this note was added to the trouble ticket
note[index]->author	Who added this note
note[index]->text	Array of strings
internalNote	Array of records (Internal notes are only available to NO and IO entity)
internalNote[index]->date	Datetime stamp for when this note was added to the trouble ticket
internalNote[index]->author	Who added this note
internalNote[index]->text	Array of strings
subState	Possible values are: NEW: Initial subState for all tickets SCHEDULED: Only for SAWA tickets. Set by system when appointment is scheduled. SP must now assign ticket to IO entity. REJECTED: REMEDIED: Optional subState, which can be used by NO/IO to signal the issue has been resolved, but they have not fully completed there work. RESOLVED: Issue/Required work has been completed. Ticket must be assigned back to SP entity. RETURNED: CANCELLED: Set by SP's, if a ticket is nolonger relevant. REOPENED: Set by SP's, is a ticket passed back to them needs further work. Ticket must again for STAN tickets be assigned to NO entity and for SANA tickets be assigned to IO. For SAWA tickets an appointment must be scheduled. REOPN_SCHD: Only for SAWA tickets. Set by system when appointment is scheduled. SP must assign ticket to IO once appointment has been scheduled.
expectedOnsiteStart	Datetime stamp for when a technician is expected to visit the end customer. This is the beginning of the assigned timeslot, not the actual arrival time.
expectedOnsiteEnd	Datetime stamp for when a technician is expected to visit the end customer. This is the end of the assigned timeslot, not the actual time of completion.
startSLA	Set to same datetime as creationDate, unless classification is upgraded. In these circumstances, the NO/IO is allowed to move startSLA to the time of the upgrade.
remoteStarted	When did the NO/IO start remote work
suspendedSLA	In case NO/IO find them self waiting for the end customer, before they can start onsite work, they are allowed to temporarily susped the SLA. This field will hold the datetime stamp in case this happens.

resumedSLA	In case the SLA was suspended, this field will hold a datetime stamp for when the SLA was resumed.
onsiteStarted	When did the NO/IO start onsite work
lastUpdated	When was this trouble ticket last updated
appointment *Only included for SAWA type trouble tickets*	Record
appointment->date *Only included for SAWA type trouble tickets*	The selected appointment date: yyyy-MM-dd Ex. 2021-03-25
appointment->slotCd *Only included for SAWA type trouble tickets*	Appointment Date Slot (AM_NO, AM_PR_1, AM_PR_2, PM_NO, PM_PR_1, PM_PR_2, PM_PR_3)
errorAfterInstallation Available from API version 2.0 depending on config	errorAfterInstallation property [Boolean] will be true, if the Trouble Ticket has been created within the first n working days after initial order completion. The n number of working days is defined in the specific Service Provider Agreement. For existing partners this property will be hidden by default to avoid breaking the integrations. If a partner wish to make use if this feature, raise a General IT Request to OpenNet via the OpenNet Partnerportal.
contactEmail *Available from API version 2.0*	Contact E-mail for reference. If a trouble ticket was created using API version 1.0 this field will hold the value "N/A".
relatedEntities->serviceProvider Available from API version 3.0	Allows the caller to identify which Service Provider entity is related to the Trouble Ticket. Note that the value is a string, specifying the entity code. Ex. "SP02" or similar.
relatedEntities->infraStructureOwner Available from API version 3.0	Allows the caller to identify which Infrastructure Owner entity is related to the Trouble Ticket. Note that the value is a string, specifying the entity code. Ex. "IO02" or similar. This information will allow NO's, working for multiple IO's, to easier identify which IO is related to the specified Trouble Ticket.
relatedEntities->networkOperator Available from API version 3.0	Allows the caller to identify which Network Operator entity is related to the Trouble Ticket. Note that the value is a string, specifying the entity code. Ex. "NO02" or similar.

Table 29: Data items - getTroubleTicket

11.3.2 Update Trouble Ticket

Whilst the trouble ticket is in status open parties associated to the ticket can add notes, change subState, classification etc.. Updates to classification, subState are logged using system created notes. Notes will always include the author and a timestamp.

The sequence diagram for updateTroubleTicket is shown in the figure below**:**

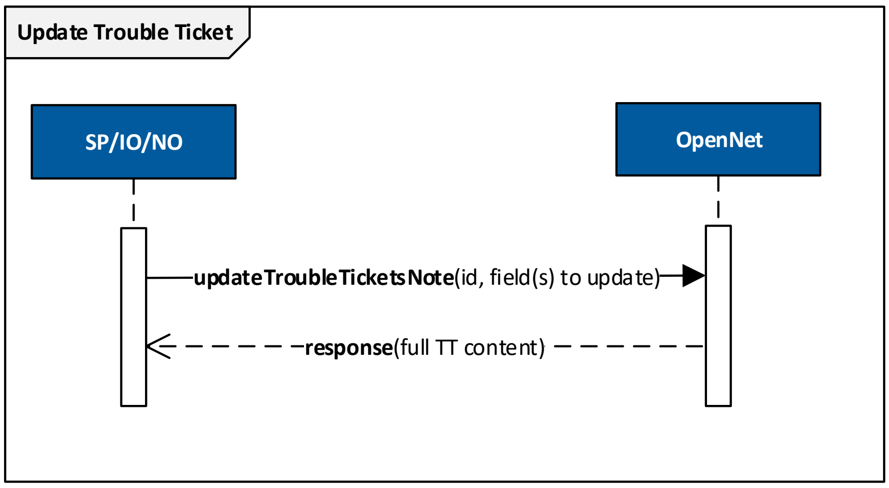

Update Trouble Ticket
SP/IO/NO
OpenNet
updateTroubleTicketsNote(id, field(s) to update)
response(full TT content)

Figure 22: Update Trouble ticket

11.3.2.1 Data Definition

The following provides a high-level view of the data items expected to support the INPUT and RESPONSE payload. This data is subject to change as design/development evolves through. Id value is a required information, but the other fields are optional and can be used in any combination needed.

Operation: updateTicket

Input - Data Item	**Description**
Id	Trouble Ticket ID
note	Array of free text fields to include detail. Can hold multiple lines/elements.
internalNote	Array of free text fields to include details. Can hold multiple lines/elements. Internal notes are only available to NO and IO entities, and will not be visible to related SP entity.
assignee	Include when reassign of trouble ticket is required.
assignee->type	When assigning serviceSubscriptionID related incidents around, assignee type "ENTITY" must be used to route a Trouble Ticket to the correct entity.
assignee->value	For assignee type = ENTITY this value must be "NO" or "IO".
contactName	Contact name for reference
contactNumber	Contact number for reference
contactEmail	Contact e-mail for reference
subState	See description of Trouble Ticket Sub State below.
classification	For STAN trouble tickets classification code can be "MAJ" = Major "MIN" = Minor "IMI" = Immediate Service (Only allowed for H1 and H3 services) "SER" = Service Request (Only account level trouble tickets. Not in use currently.) "OTH" = Other (Not in use) For serviceSubscriptionID related incidents use "MAJ", "MIN". For H1/H3 services "IMI" is also possible. For SANA and SAWA trouble tickets, SP/NO/IO are not expected to reclassify a ticket once created.

externalIdSP	Free text field, that can be freely used by the SP.
Response	
	Full ticket details

Table 30: Data items - updateTicketsNotes

11.3.3 Get All Trouble Tickets Assigned To User

The Get All Trouble Tickets Assigned To User service allows the Service Provider to retrieve an array of Trouble Ticket ID's currently assigned to the caller.

The sequence diagram for getAllTroubleTicketsAssignedToUser is shown in the figure below**:**

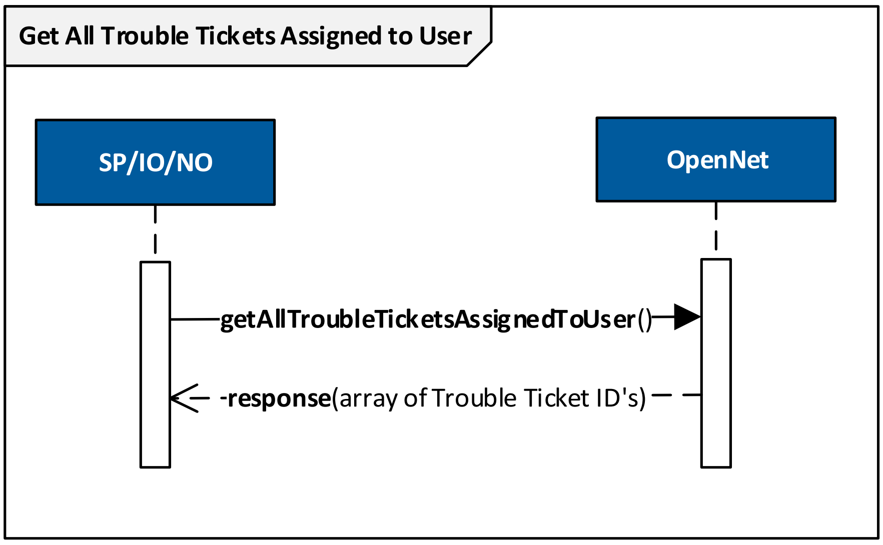

Get All Trouble Tickets Assigned to User
SP/IO/NO
OpenNet
getAllTroubleTicketsAssignedToUser()
-response(array of Trouble Ticket ID's)

Figure 23: Get All Trouble Ticket Assigned to User

11.3.3.1 Data Definition

The following provides a high-level view of the data items expected to support the INPUT and RESPONSE payload. This data is subject to change as design/development evolves through.

Operation: getTroubleTicket

Input - Data Item	**Description**
Response	
troubleTicketIds	Array of Trouble Ticket Id's assigned To User

Table 31: Data items - Get All Trouble Tickets Assigned To User

11.3.4 Search for Open Trouble Tickets

The Trouble Ticket Search API, allows the caller to get a list of trouble ticket ID’s which match the search parameters. The use of this API, will allow the caller to only fetch the full trouble ticket dataset, for trouble tickets which has been updated after a specific timestamp and either all types or only a specific type of trouble tickets. The type specifier is only relevant for entities using the SANA and SAWA trouble ticket types on top of the normal STAN trouble ticket type.

The sequence diagram for searchOpenTroubleTickets is shown in the figure below**:**

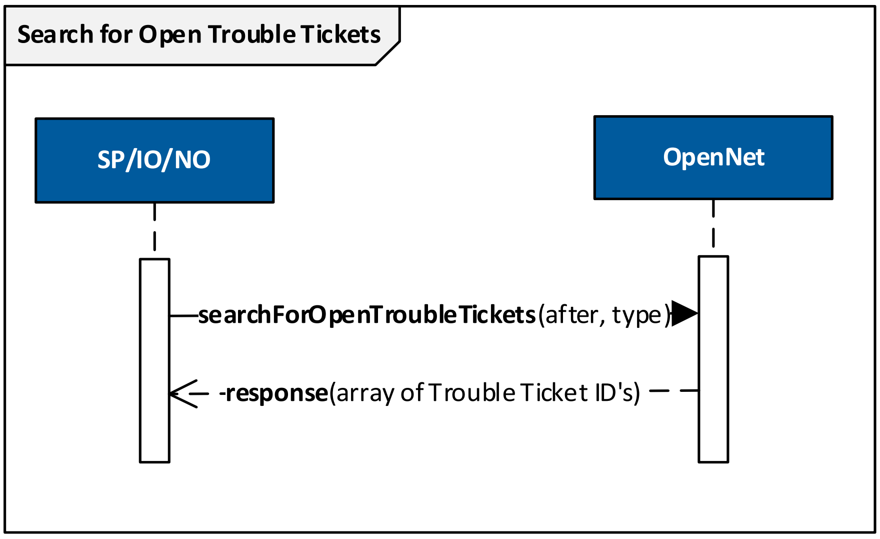

Search for Open Trouble Tickets
SP/IO/NO
OpenNet
searchForOpenTroubleTickets(after, type)
-response(array of Trouble Ticket ID's)

Figure 24: Search for Open Trouble Tickets

11.3.4.1 Data Definition

The following provides a high-level view of the data items expected to support the INPUT and RESPONSE payload. This data is subject to change as design/development evolves through.

Operation: searchForOpenTroubleTickets

Input - Data Item	**Description**
after (query param)	If specified only the ID's of trouble tickets where lastUpdated property is larger than the specified timestamp will be returned. The parameter will be in UTC timestamp and must be written in the following format: yyyyMMddThhmmssZ Example: 20210429T145345Z If not specied all trouble tickets with status "OPEN" will be in scope of the search.
type (query param)	If specified only trouble tickets of the specified type will be included in the search result. Possible values are: STAN, NETW, SANA, SAWA and ALL If either type=ALL or type value is not specified in the call, all trouble tickets, regardless of their type, will be included in the scope of the search.
status (query param)	If specified trouble tickets which match the specified status will be included in the search result. Possible values are: OPEN, CLOSED, ALL If status query parameter is not defined, then system will default to returning only trouble tickets with status=OPEN. status=ALL will result in search result including both trouble tickets with status=OPEN and CLOSED.
Response	
troubleTicketIds	Array of "OPEN" Trouble Ticket Id's which are related to the user

Table 32: Data items – Search for Open Trouble Tickets

11.3.5 How to Use the Trouble Ticket System for serviceSubscriptionID Related Incidents

The OpenNet trouble ticket system utilize a subState parameter, which is used to coordinate the communication between SP's, NO's and IO's. OpenNet enforce a subState graph as visualized in *Figure 25: Trouble Ticket Sub State*.

General rules when interacting with trouble tickets:

1. SP's are allowed to add notes to a trouble ticket any time, but NO/IO will only collect these if a trouble ticket is assigned to their entity.
2. When a trouble ticket is assigned to an entity (SP, NO or IO), this entity is allowed to manipulate fields under their control. See
3. SP's are allowed to change a trouble ticket subState to CANCELLED, when a trouble ticket is not assigned to the SP, as long as the state graph allow the transition. Doing so shall result in NO/IO cancelling scheduled work in their systems and assign the trouble ticket back to SP.
4. SP's are only allowed to close trouble tickets that are assigned to them, i.e. SP need to wait for NO/IO to assign the trouble ticket back to SP before closing.
5. If a SP reopens a trouble ticket, a note must always be added to tell the NO/IO why this is the case. SP's must also remember to reassign the trouble ticket to the south bound entity. For STAN trouble tickets this is always the NO entity. For SANA and SAWA trouble tickets this is always the IO entity.
6. Before SP's can assign SAWA trouble tickets to IO it must be scheduled. This is the case for both NEW and REOPENED trouble tickets. subState will change to SCHEDULED and REOPENED_SCHEDULED respectively.

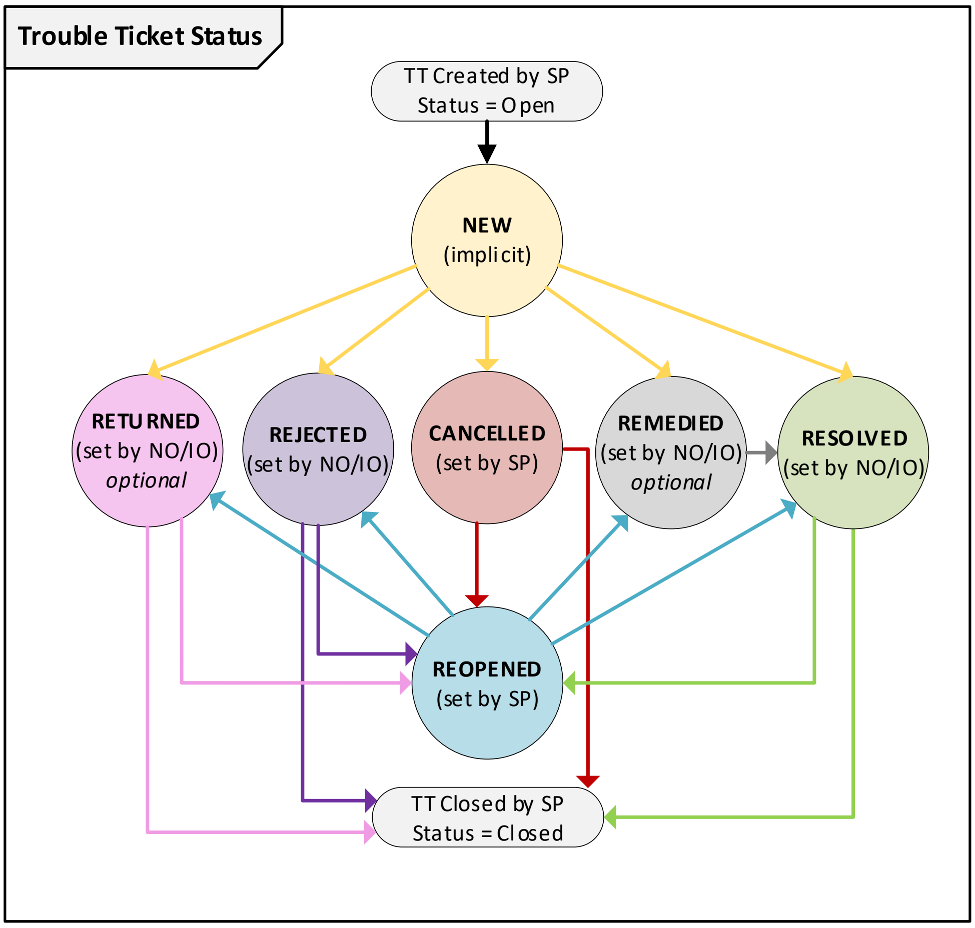

Trouble Ticket Status
TT Created by SP Status = Open
NEW (implicit)
RETURNED (set by NO/IO) optional
REJECTED (set by NO/IO)
CANCELLED (set by SP)
REMEDIED (set by NO/IO) optional
RESOLVED (set by NO/IO)
REOPENED (set by SP)
TT Closed by SP Status = Closed

Figure 25: Trouble Ticket Sub State (Note that SCHEDUED and REOPENED_SCHEDULED subStates are not included in the graph at this time)

Trouble Ticket Field	Field Owner (What entity writes to this field)	Allow manipulation when not assigned to writing entity
Classification	SP	Yes
outageId	NO/IO	No
externalIdSP	SP	Yes – Not used by NO/IO
externalIdNO	NO	Yes – Not used by SP/IO
externalIdIO	IO	Yes – Not used by NO/SP
Description	SP	No
severity	Depends on trouble ticket type. Type=STAN: Value is controlled by OpenNet Type=SANA: Value is set when SP creates Trouble Ticket. Type=SAWA: Value is set when SP creates Trouble Ticket.	No
Type	SP	Fixed value
serviceSubscriptionId	SP	Set on create trouble ticket
accountNo	OpenNet	N/A
contactName	SP, NO, IO	No
contactNumber	SP, NO, IO	No
activityTask	OpenNet	Not in use
assignee->type	SP, NO, IO	No
assignee->value	SP, NO, IO	No
Id	OpenNet	N/A
Status	SP	No
creationDate	OpenNet	N/A
note[index]->date	OpenNet	N/A
note[index]->author	OpenNet	N/A
note[index]->text	SP, NO, IO	Yes
subState	SP, NO, IO	No – Exception is CANCELLED that can be set by SP when subState graph allows this
expectedOnsiteStart	NO, IO	No
expectedOnsiteEnd	NO, IO	No
startSLA	OpenNet, NO, IO	No
remoteStarted	NO, IO	No
suspendedSLA	NO, IO	No
resumedSLA	NO, IO	No
onsiteStarted	NO, IO	No
lastUpdated	OpenNet	N/A

Table 33: Trouble Ticket Field Owner

11.4 Planned & Unplanned Outages

The OpenNet system provides an ability for NO/IO to create planned and unplanned outage notifications to relevant SP's based on affected services.

Services will be provided for the following:

- Create Outage
- Update Outage
- Upload Related Service Subscription ID's (Update current list of related Service Subscription ID's)
- Get Upload Status (When uploading releated Service Subscription ID's)

11.4.1 Create Outage

The Create Outage service allows the NO/IO to create an outage if a specific type (PLANNED/UNPLANNED), and have the outage ID returned.

The sequence diagram for Create Outage is shown in the figure below**:**

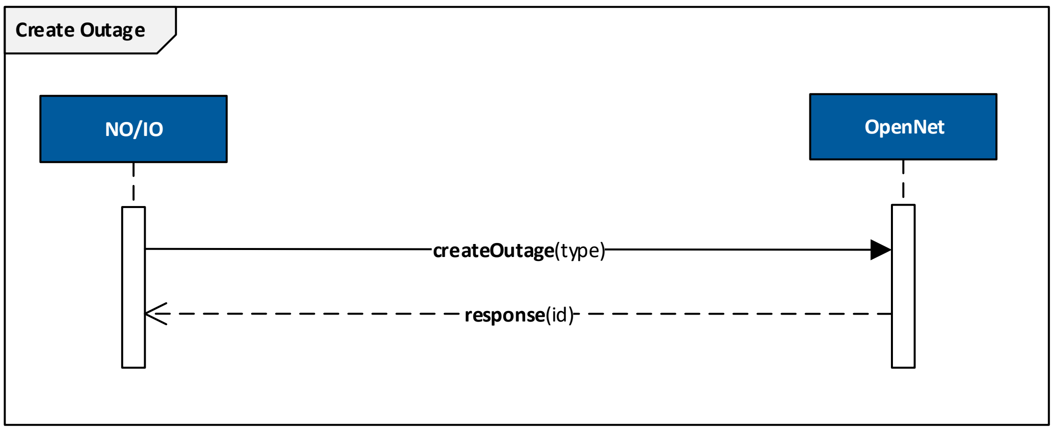


Figure 26: Create Outage

11.4.1.1 Data Definition

The following provides the data items expected to support the REQUEST and RESPONSE payload.

Operation: createOutage

Input - Data Item	**Required**	**Description**	**Data type**
type	Yes	Value can be: "PLANNED" or "UNPLANNED"	String
Response (http response 200 – OK)			
id		The outage ID	Int64
Response (http response 400 – Bad request)			
responseCode		Error classification: "FAILURE"	String
message		Description of the failure reason	
Response (http response 500 – Internal server error)			
		Outage was not created due to an internal server error.	

Table 34: Data items – createOutage

11.4.2 Update Outage

The Update Outage service allows NO/IO to update all meta data, which form the content of an outage record. Fields must be published when possible, i.e. an expectedServiceImpactEnd must be set for an unplanned outage when possible and updated at the outage progress. Regular updates are also expected for unplanned outages, in the form of progress notes.

The sequence diagram for Update Outage is shown in the figure below**:**

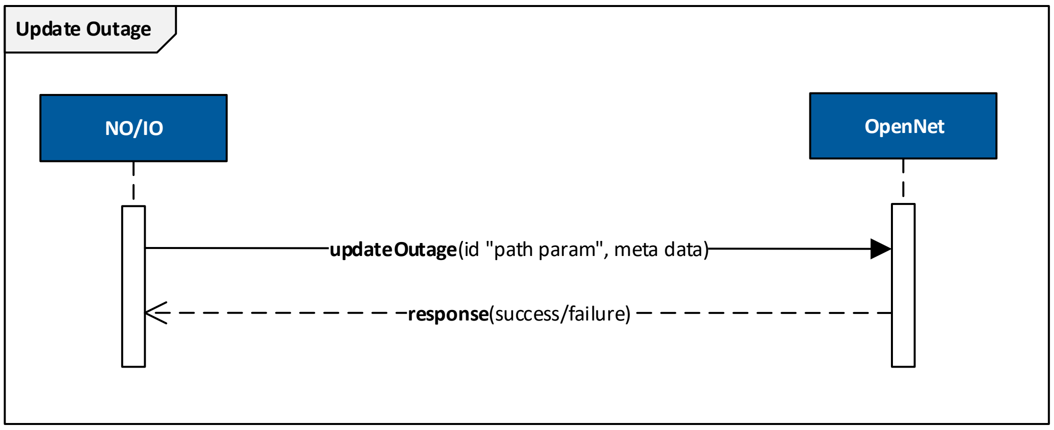

Update Outage
NO/IO
OpenNet
updateOutage(id "path param", meta data)
response(success/failure)

Figure 27: Update Outage

11.4.2.1 Data Definition

The following provides the data items to support the REQUEST and RESPONSE payload.

Operation: updateOutage

Input - Data Item	**Required**	**Description**	**Data type**
id [path param]	Yes	The Outage ID. Note this is a path parameter.	
status	No	Possible values are "OPEN", "WORKING", "SUSPENDED", "CLOSED", "CANCELLED". For both Planned and Unplanned outages, a status change to CLOSED or CANCELLED is expected as last update.	String
workStart	No	When the work is expected to start. E.g. "2018-09-04T22:00:00+02:00"	String
serviceImpactStart	No	When is the work expected to impact services. E.g. "2018-09-04T22:00:00+02:00"	String
expectedServiceImpactEnd	No	When is the service impact expected to end. E.g. "2018-09-04T23:00:00+02:00"	String
expectedWorkEnd	No	When is this outage expected to be resolved/finished. E.g. "2018-09-04T23:00:00+02:00"	String
serviceImpactCode	No	Code defining what impact the service will have. Expected code: OTHER, DOWN, ERRATIC, IMPAIRED.	String
Description	No	A free text description. Used as a header/title for this outage. This is stored in a single field, which will be updated if written multiple times for the same outage.	String
Note	No	More details and progress updates. Notes are stored in an array structure with timestamps.	String
Response (http response 204 – No Content)			

Response (http response 400 – Bad request)			
responseCode		Error classification: "FAILURE"	String
message		Description of the failure reason	String
Response (http response 500 – Internal server error)			
		Outage was updated due to an internal server error.	

Table 35: Data items – updateOutage

11.4.3 Upload Related Service Subscription ID's

The request is used to update the list of outage related serviceSubscriptionID's. The update request should hold all related serviceSubscriptionID's. Logic to handle changing if a serviceSubscriptionID is currently related or previously related to an outage is handled within OpenNet. The completion of the upload process, must be fetched using getUploadStatus API.

The sequence diagram for Upload Related Service Subscription ID's is shown in the figure below**:**

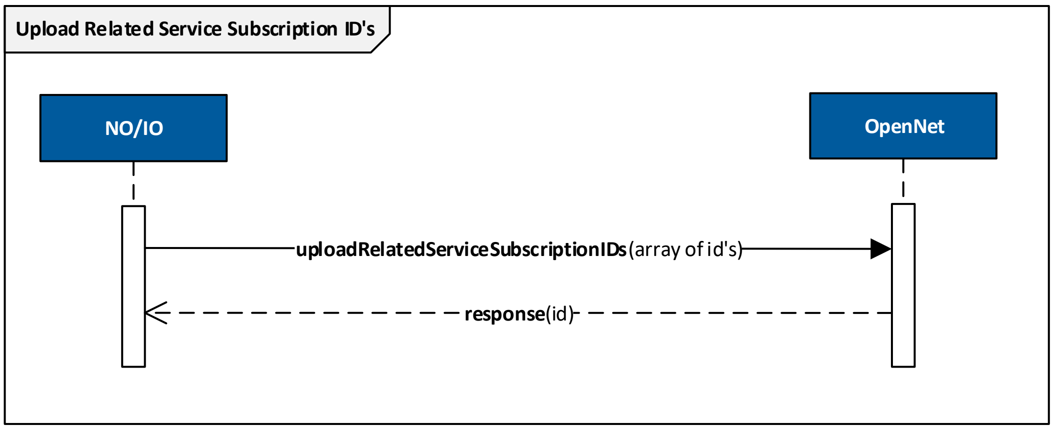

Upload Related Service Subscription ID's
NO/IO
OpenNet
uploadRelatedServiceSubscriptionIDs(array of id's)
response(id)

Figure 28: Upload Related Service Subscription ID's

11.4.3.1 Data Definition

The following provides the data items expected to support the REQUEST and RESPONSE payload.

Operation: uploadRelatedServiceSubscriptionIDs

Input - Data Item	**Required**	**Description**	**Data type**
id [path param]	Yes	The Outage ID. Note this is a path parameter.	String
ids	No	Array of related service subscription ID's. If not specified/null, system will regard it as no Service Subscription ID's are related to the outage after the update.	Array of strings
Response (http response 200 – OK)			
id		Upload ID. This must be used as input to getUploadStatus API.	Int64
Response (http response 400 – Bad request)			
responseCode		Error classification: "FAILURE"	String

message		Description of the failure reason	string
Response (http response 500 – Internal server error)			
		Outage was not created due to an internal server error.	

Table 36: Data items – Upload Related Service Subscription ID's

11.4.4 Get Upload Status

The Get Upload Status service allows the NO/IO get the status of a previously initiated upload of related Service Subscription ID's.

The sequence diagram for Create Outage is shown in the figure below**:**

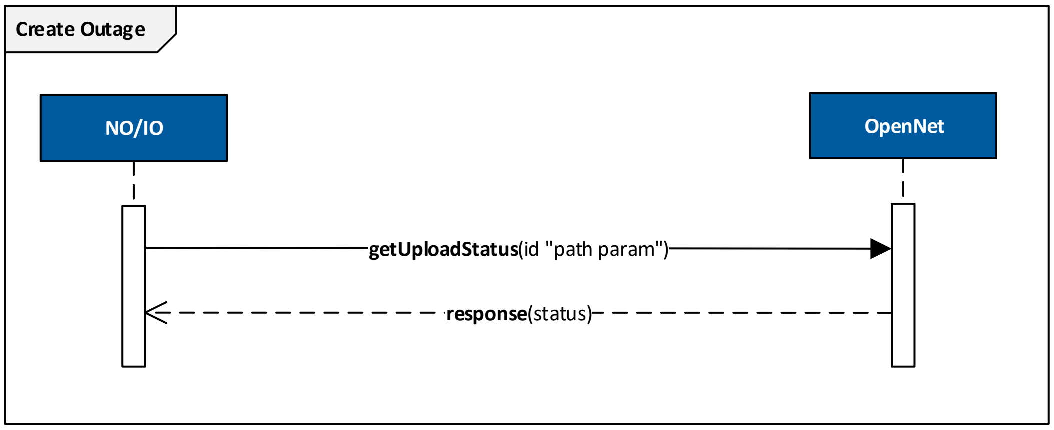


Figure 29: Get Upload Status

11.4.4.1 Data Definition

The following provides the data items expected to support the REQUEST and RESPONSE payload.

Operation: getUploadStatus

Input - Data Item	**Required**	**Description**	**Data type**
id [path param]	Yes	The Upload ID. Note this is a path parameter.	String
Response (http response 200 – OK)			
id		The Upload ID	Int64
status		Status of the previously initiated upload process. Possible values are: COMPLETE, INCOMPLETE, FAILURE	String
message		If status="FAILURE" this message will state what the error is	String
Response (http response 400 – Bad request)			
responseCode		Error classification: "FAILURE"	String
message		Description of the failure reason	String
Response (http response 404 – Not found)			
		The upload ID was not found	
Response (http response 500 – Internal server error)			
		Outage was not created due to an internal server error.	

Table 37: Data items – getUploadStatus

12. Timeslot Management

OpenNet has taken a semi-online approach to Timeslot Management, which consist of multiple calendars, which are stored locally within OpenNet. The IO's will have full control over the number of bookings that can be made within each of their calendars, i.e. the number of points which will be available within each timeslot using the Timeslot Management API's.

When a SP is required to book an appointment, the SP will ask OpenNet for a list of timeslots with available capacity, and based on address information from the IO, OpenNet will look in the calendar for the correct IO+Work Field Area and provide the requested information to the SP. The earliest date returned to the SP can never be before the returned total service delivery lead time which is based on information in the Address File and the result of the capacityCheckAndReserve request sent to the NO during the order capture process.

The detailed planning, which will take the driving time of the technian and different resource needs into account, can be managed by either IO manipulating the amount of allowed bookings within a timeslot if a complex order is pushed through, or by IO doing a percentage calculation and thereby limiting the total amount of allowed bookings to take the more complex orders into account.

12.1 Overview of Technician Calender

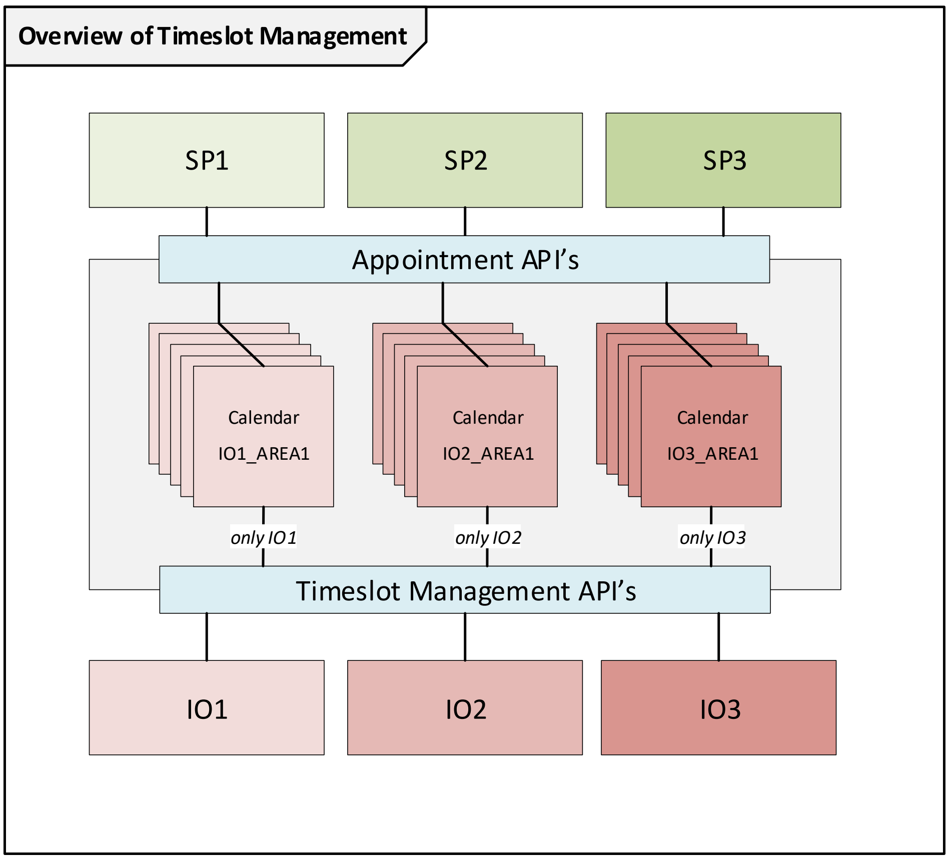


Figure 30: Technician Calender Overview

12.2 Data Segregation

An IO can only access their own calendars within the OpenNet system. SP's can, based on their specific order, get information from the calendar relevant for the order.

12.3 Calendar Data

Property name	**Data type**	**Example values**	**Description**
date	date	"2018-10-25"	The date format is yyyy-MM-dd.
infrastructureOwnerCode	string	"IO01"	Internal marking to separate calendars from each IO.
workFieldArea	string	"AREA1" "AREA2"	Each IO has the possibility to separate their infrastructure into geographical areas. This can be used to handle areas that has limited access, areas that has longer driving time for the technicians and generally gives the IO some flexibility in how technician capacity is geographically distributed.
closed	boolean	true false	If this value is true, no more bookings will be allowed from the SP's in the specific area within the timeslot. IO can open/close days or go down to an date+area+
totalAvailablePoints	integer	10	Total amount of bookings allowed within the specified timeslot.
workType	string	"INSTALL" "TROUBLE"	The calendar system within OpenNet core software, has been prepared to handle multiple types of work, but currently only "INSTALL" type is in use for actual bookings.
workTimeSlotCd	string	"AM_NO" (Normal slot before noon) "AM_PR_1", "AM_PR_2", (Premium slot before noon) "PM_NO" (Normal slot afternoon) "PM_PR_1", "PM_PR_2", "PM_PR_3" (Premium slot afternoon) Legacy codes AM_PR and PM_PR may exist for some IO's.	To allow some flexibility within the commercial agreements, the OpenNet system use codes and not specific times when specifying timeslots.
allocatedPoints	integer	7	The amount of bookings currently allocated in the specified timeslot. When a SP booking is made allocatedPoints will be incremented. If a SP booking is cancelled or rescheduled to another date/time, allocatedPoints will be decremented by one point.

			The decrement value can only be configured by OpenNet based on SP and main product (H1/H2/H3).

Table 38: Calendar data

12.4 Timeslot Management Services

A total of four services will be available to get and update information within the calendars available to the IO:

- getTimeslotData
- close
- open
- updateTimeslotData

12.4.1 Get Timeslot Data

This service can be used to get timeslot information from the OpenNet system.

12.4.1.1 Data Definition

Operation: getTimeslotData

Input - Data items	**Description**
date	Date for which to return timeslots. Date format is yyyy-MM-dd. Example: "2018-10-28"
workFieldArea	**Optional** field that can be used to narrow down the result data set. Example: "AREA1"
Response(http response code 200)	**Description**
timeslots	Array structure that holds all returned timeslots
timeslots[index]->workFieldArea	Specify the area covered by this timeslot. Example: "AREA1"
timeslots[index]->closed	If true no bookings can be done in this slot.
timeslots[index]->totalAvailablePoints	Total amount of allowed bookings within this timeslot. Example: 15
timeslots[index]->workType	Currently only "INSTALL" is used.
timeslots[index]->workTimeSlotCd	Specify what timeslot this record is linked to. Example: "AM_NO"
timeslots[index]->allocatedPoints	How many bookings have been done in this timeslot. Example: 7
Response(http response code 400)	**Description**
responseCode	Example: "FAILURE"
Message	Description of the FAILURE reason.
Response(http response 500)	**Description**
Null	

Description	**JSON**
Request	{
	"date": "2018-10-28",
	"workFieldArea": "AREA1"
	}
Response	{
	"timeslots": [
	{
	"workFieldArea": "AREA1",
	"closed": false,
	"totalAvailablePoints": 12,
	"workType": "INSTALL",
	"workTimeSlotCd": "AM_NO",
	"allocatedPoints": 5
	},
	{
	"workFieldArea": "AREA1",
	"closed": false,
	"totalAvailablePoints": 10,
	"workType": "TROUBLE",
	"workTimeSlotCd": "AM_NO",
	"allocatedPoints": 0
	},
	{
	"workFieldArea": "AREA1",
	"closed": false,
	"totalAvailablePoints": -12,
	"workType": "INSTALL",
	"workTimeSlotCd": "AM_PR_1",
	"allocatedPoints": 0
	},
	{
	"workFieldArea": "AREA1",
	"closed": false,
	"totalAvailablePoints": 10,
	"workType": "TROUBLE",
	"workTimeSlotCd": "AM_PR_1",
	"allocatedPoints": 0
	},
	{
	"workFieldArea": "AREA1",
	"closed": false,
	"totalAvailablePoints": 12,
	"workType": "INSTALL",
	"workTimeSlotCd": "PM_NO",
	"allocatedPoints": 7
	},
	{
	"workFieldArea": "AREA1",
	"closed": false,
	"totalAvailablePoints": 12,
	"workType": "TROUBLE",
	"workTimeSlotCd": "PM_NO",
	"allocatedPoints": 0
	},
	{
	"workFieldArea": "AREA1",
	"closed": false,
	"totalAvailablePoints": 12,
	"workType": "INSTALL",

	"workTimeSlotCd": "PM_PR_1",
	"allocatedPoints": 0
	},
	{
	"workFieldArea": "AREA1",
	"closed": false,
	"totalAvailablePoints": 10,
	"workType": "TROUBLE",
	"workTimeSlotCd": "PM_PR_1",
	"allocatedPoints": 0
	}
	]
	}

Table 39: Data definition

12.4.2 Update Timeslot Data

This service is used to update timeslot data on an array of timeslot items to update.

Operation: updateTimeslotData

Input - Data items	**Description**
searchDate	The date on which the specified slots should be updated. Date format is "yyyy-MM-dd" Example: "2018-10-28"
timeslots	Array of timeslots to update
timeslots[index]->workFieldArea	Required field which specify the Work Field Area. Example: "AREA4"
timeslots[index]->closed	If not included, slot will be opened i.e. value default to false. This is a boolean value. Example: "false"
timeslots[index]->totalAvailablePoints	If not included value defaults to 0 points.
timeslots[index]->workType	Required field which specify the workType. Currently only workType "INSTALL", will be used for actual appointment bookings.
timeslots[index]->workTimeslotCd	Required field that specify the exact parameter.
Response(http responsecode 204)	**Description**
Null	Specified timeslots have been updated.
Response(http responsecode 400)	**Description**
responseCode	Example: "FAILURE"
Message	Description of the FAILURE reason.
Response(http response 500)	**Description**
Null	Timeslot data could not be updated due to internal server error

Table 40: Data definition

12.4.3 Close Timeslots

This service is used to close timeslots for further bookings narrowed down from a full day (date) using the optional fields workFieldsArea, workType and/or workTimeSlotCd.

Operation: closeTimeslots

Input - Data items	**Description**
date	Required field defining what date is manipulated. Date format is "yyyy-MM-dd" Example: "2018-10-28"
workFieldArea	Optional field defining which area to close. Example: "AREA1"
workType	Optional field defining which work type to close. Example: "INSTALL"
workTimeSlotCd	Optional field defining which time slot to close. Example: "AM_NO"
Response(http responsecode 204)	**Description**
Null	Specified timeslots have been closed
Response(http responsecode 400)	**Description**
responseCode	Example: "FAILURE"
Message	Description of the FAILURE reason.
Response(http responsecode 500)	**Description**
Null	Timeslot data could not be updated due to an internal server error

Table 41: Data definitions

12.4.4 Open Timeslots

This service is used to open timeslots for bookings if points are available, narrowed down from a full day (date) using the optional fields workFieldsArea, workType and/or workTimeSlotCd.

Operation: openTimeslots

Input - Data items	**Description**
date	Required field defining what date is manipulated. Date format is "yyyy-MM-dd" Example: "2018-10-28"
workFieldArea	Optional field defining which area to open. Example: "AREA1"
workType	Optional field defining which work type to open. Example: "INSTALL"
workTimeSlotCd	Optional field defining which time slot to open. Example: "AM_NO"
Response(http responsecode 204)	**Description**

Null	Specified timeslots have been open
Response(http responsecode 400)	**Description**
responseCode	Example: "FAILURE"
Message	Description of the FAILURE reason.
Response(http response 500)	**Description**
Null	Timeslot data could not be updated due to an internal server error

Table 42: Data definitions

13. Exceptions

In the event that an exception occurs while processing a request to the OpenNet Integration Layer, a response will be returned to the service consumer. When ever possible this response will contain information detailing why the request could not complete successfully.

For successful responses OpenNet Integration Layer uses RESTful interface status standards which includes positive and negative status code/numbers. For more detail, see: http://www.restapitutorial.com/httpstatuscodes.html.

13.1 Standard Exceptions

A set of errors can occur in any request to OpenNet, these are named Standard Exceptions. The errors are:

- Server Error
- Rate Lime Exceeded
- Forbidden
- Bad Request (see below)

13.2 Example Bad Request

If OpenNet cannot understand a request due to invalid parameters, it will return a bad request exception. This is analogous to the HTTP 400 error.

If encountering this, check the response body for more information. In the event an exception occurs that the system does not expect, you will be guided to contact OpenNet support

14. IO Charging file

During the delivery of a Service Providers end-customer service, it may be necessary to perform specific network orientated tasks. In specific cases the task may result in effort/cost that needs to be passed on to the Service Provider.

The Infrastructure Owner will provide a file defining transactions which must be applied to a specific service (i.e. serviceSubscriptionID), which is subsequently applied to the Service Providers account for charging/billing.

14.1 File name and location

Folder on OpenNet sFTP server: /adjin

File name format: ADJ_LOAD_yyyy_MM_dd_xxxxxx.csv
Example: ADJ_LOAD_2020_01_27_000001.csv

Note: "xxxxxx" represent a sequence number within a specific date, to ensure unique filenames.

14.2 File content specification

File will be in csv format with comma as text delimiter and double quotes as text qualifier. Please note that the first row should **NOT** contain colomn names, and contains the first transaction to import.

File encoding is UTF-8 without BOM.

Colomn name	**Value description**
Service Subscription ID	Id of the service to which the transaction relate
Amount Before Tax	Total amount before tax. This is always the total amount, i.e. for digging 10 meters this will hold total amount for the full 10 meters, not the price for 1 meter. See Note field regarding the quantity. The value must use comma "," as decimal separator. Dot "." is not allowed.
Nominal Code	This value must be "NOM-DDEBIT"
Record content	This value must be "D"
Sales Tax Code	This value must be "DKS"
Transaction Effective Date	Effective date for a transaction. Format: yyyy_MM_dd
Note	Free text field of maximum 80 charactors further elabrorating on the specific transaction.
Transaction Code	Short form code, to enable easier IT handling. Ex. CHEXDI, CHTVIV etc.. Codes will be introduced over time, as they become required. Generic codes are in place to handle miscs. transactions.

14.3 Sample line

"1000071617","9999,00","NOM-DDEBIT","D","DKS","2020_01_23","10m","CHEXDI"

14.4 List of transaction codes

Transaction Code	Transaction Dec
CHABSO	Alt box solution
CHAFLH	Alt fiber box loc
CHAFLL	Alt fiber box loc (light)
CHEFTC	Extra tube or cab (per m)
CHEIBT	Extended inst (BB and TV)
CHEITV	Extended inst (TV)
CHEWFC	Extended Wi-Fi coverage
CHEXDI	Extra digging (per m)
CHHRAD	RPH-Admin
CHHRIN	RPH-Instl
CHHRIO	RPH-Instl OT
CHHRNT	RPH-Netwk techn

CHHRPM	RPH-Project Mgmt
CHHRST	RPH-Spec techn
CHMATE	Materials
CHMISC	Miscellaneous Charge
CHPAPO	Patching port
CHRDFE	Repair dam fiber or equip
CHREDF	Repair dam fiber
CHREFB	Relocate fiber box
CHSTUS	Tech unjustified on site
CHTRAN	Transport fee
CHTVIV	Technician visit in vain

14.5 Loading process and error handling

A bulk load is executed each workday by OpenNet. As files are processed a file is placed in "/adjrej" with the original filename prefixed with "REJECT_". Please note that file extention will change to ".txt".

If no errors occoured during loading, a zero byte file is placed as receipt of successful load. If record errors are found, transactions that have failed, will be listed in the file placed in "/adjrej" folder together with an error description.

15. Environments

OpenNet have two environments (Pre-Production and Production) available to external partners. Depending on timing Pre-Production will hold either the same implementation as Production or include the upcoming features ready for test.

16. Endpoint URL's

16.1 Authentication API

Pre-Prod URL: **https://test.opennet.nu:8543/auth/realms/cerillion/protocol/openid-connect/token**
Production URL: **https://prod.opennet.nu:8543/auth/realms/cerillion/protocol/openid-connect/token**

16.2 Other API's

he base URL for all other API's are:
Pre-Prod: **https://test.opennet.nu:8443/apiman-gateway/cerillion**
Production: **https://prod.opennet.nu:8282/apiman-gateway/cerillion**

Request name	**Type**	**URI**	**Note**
Capacity Reservation Timed Out	DELETE	/capacityReservationTimedOut/1.0 **(Obsolete)**	**New version 2.0 deployed. Migration to using version 2.0 should be done no later**

			than July 31st 2020.
Capacity Reservation Timed Out	DELETE	/capacityReservationTimedOut/2.0	Only URI change, required due to other internal changes.
Get Fulfillment Request	POST	/fulfillment/1.0/collect **(obsolete)**	**New version 2.0 deployed. Migration to using version 2.0 should be done no later than July 31st 2020.**
Get Fulfillment Request	POST	/fulfillment/2.0/collect **(Obsolete)**	**New version 3.0 deployed on September 1st 2021. Migration to using version 3.0 within 6 months.** Add's contactEmail
Get Fulfillment Request	POST	/fulfillment/3.0/collect	Add's priority property Results are returned using a FIFO principle.
Complete Fulfillment Request	PATCH	/fulfillment/1.0 **(Obsolete)**	**New version 2.0 deployed. Migration to using version 2.0 should be done no later than July 31st 2020.**
Complete/Acknowledge Fulfillment Request	PATCH	/fulfillment/2.0**(Obsolete)**	**New version 3.0 deployed on September 1st 2021. Migration to using version 3.0 within 6 months.** No change from API version 1.0
Complete/Acknowledge Fulfillment Request	PATCH	/fulfillment/3.0	No change from API version 2.0
Add Note	POST	/basket/1.4/note **(Obsolete)**	**New version 2.0 deployed. Migration to using version 2.0 should be done no later**

			than July 31st 2020.
Add Note	POST	/basket/2.0/note **(Obsolete)**	**New version 3.0 deployed September 1st 2021.** Only URI change, required due to other internal changes.
Add Note	POST	/basket/3.0/note	Only URI change, required due to other internal changes.
Get Next Event	GET	/event/1.0/nextevent **(Obsolete)**	**New version 2.0 deployed. Migration to using version 2.0 should be done no later than November 1st 2023.**
Update Event	PATCH	/event/1.0/updateevent **(Obsolete)**	**New version 2.0 deployed. Migration to using version 2.0 should be done no later than November 1st 2023.**
Create Event	POST	/event/1.0 **(Obsolete)**	**New version 2.0 deployed. Migration to using version 2.0 should be done no later than November 1st 2023.**
Get Next Event	GET	/event/2.0/nextevent	Only URI change, required due to other internal changes.
Update Event	PATCH	/event/2.0/updateevent	Only URI change, required due to other internal changes.
Create Event	POST	/event/2.0	Only URI change, required due to other internal changes.
Update UDF	POST	/service/1.1/udf	

Get All Trouble Tickets assigned to user	GET	/troubleticket/1.0/all **(Obsolete)**	**New version 2.0 deployed. Migration to using version 2.0 should be done no later than July 31st 2020.**
Get Trouble Ticket (Single one)	GET	/troubleticket/1.0 **(Obsolete)**	**New version 2.0 deployed. Migration to using version 2.0 should be done no later than July 31st 2020.**
Update Trouble Ticket	PATCH	/troubleticket/1.0 **(Obsolete)**	Covers notes, assignment etc.. **New version 2.0 deployed. Migration to using version 2.0 should be done no later than July 31st 2020.**
Get All Trouble Tickets assigned to user	GET	/troubleticket/2.0/all	**New version 3.0 deployed. Move to using this version within 6 months.** No change from API version 1.0
Get Trouble Ticket (Single one)	GET	/troubleticket/2.0	**New version 3.0 deployed. Move to using this version within 6 months.** Add's contactEmail
Update Trouble Ticket	PATCH	/troubleticket/2.0	**New version 3.0 deployed. Move to using this version within 6 months.** Covers notes, assignment etc..
Search Open Trouble Tickets	GET	/troubleticket/2.0/search	**New version 3.0 deployed. Move to using this version**

			within 6 months. Allows caller to search for only updated trouble tickets, rather than always having to fetch all tickets one at a time
Get All Trouble Tickets assigned to user	GET	/troubleticket/3.0/all	No change from API version 2.0
Get Trouble Ticket (Single one)	GET	/troubleticket/3.0	Add's information about relatedEntities to reponse
Update Trouble Ticket	PATCH	/troubleticket/3.0	Response changed from version 2.0 to match that of GET Trouble Ticket
Search Open Trouble Tickets	GET	/troubleticket/3.0/search	No change from API version 2.0
Get Timeslot data	POST	/timeslot/1.0/search	
Open (timeslot management)	POST	/timeslot/1.0/open	Covers open day optionally limited to specific workFieldArea, workType and/or workTimeSlotCd.
Close (timeslot management)	POST	/timeslot/1.0/close	Covers close day optionally limited to specific workFieldArea, workType and/or workTimeSlotCd.
Update Timeslot Data	POST	/timeslot/1.0/update	
Create Outage	POST	/outage/1.0	
Update Outage	PATCH	/outage/1.0/{outage id}	The specific outage id must be part of the path, and replaces {outage id}
Update Related Service Subscription ID's	PATCH	/outages/1.0/{outage id}/relatedServiceSubscriptionIDs	The specific outage id must be part of the path, and replaces {outage id}
Get Upload Status	GET	/outages/1.0/upload/{upload id}	The specific upload id must

			be part of the path and replaces {upload id}
Support Tools	POST	/supporttools/1.0	

Table 43: Test/Pre-production environement URL's

17. API Rate limits and quota policies

This section is subject to change, to allow OpenNet to provide the best experience to our partners, while maintaining the integrity and performance of the platform.

OpenNet use a combination of rate limits and quota policies to protect the platform from out of control code, and in some instances also to prevent specific actions from happening without prior agreement.

In the last category we find mass termination, where a high volume of services are terminated within a short period of time.

The OpenNet API's can be split into three categories. API's that are called prior to placing orders, API's only called during a well defined flow (Order capture/Change/Terminate etc.) and then API's that need to be polled with a sensible frequency (Fulfillment API's, Event APIs, Trouble Ticket related API's).

For the first two categories OpenNet set rate limits and policies based normal use. This means that abnormal use of API's like Pre-Sales Capacity Check will breach the rate limit or quota policies preventing further calls within the returned time period.

Endpoint/API group	**Limits are set based on**
Fulfilment API's	Users are to call fulfilment request collect every 2 minuttes or slower, as long as http 204 is returned. When http 200 is returned, users are to empty the queue, i.e. call fulfilment request collect until http 204 is again returned. Actual delay between checking fulfilment request collect should reflect the expected order volume. If higher volumes are not expected a 10 minutte interval is recommended.
Trouble Ticket API's	Users are to call getAllTroubleTicketsAssignedToUser every 2 minutes or slower (Be aware of SLA implications if going with slower, but quicker will result in unwanted resource usage on both sides), and register new tickets that show up in the list, because tickets has been assigned to the user. When new tickets are registered, users would also need to call getTroubleTicket for the specific ticket id to get the content first time. Users would need to call getTroubleTicket for each ticket assigned to them every 10 minuttes. SP's can provide additional information or cancel a ticket, which is why this call is needed, but only requires a low frequency update.
Event API's (Not relevant for all NO's/IO's)	Users are to call getNetEvent every 10 minutes or slower, as long as http 204 is returned. If a http 200 is returned, users are to continue to call until the queue is empty, i.e. http 204 is again returned. Rate limits are set to handle the max amount of events generated towards a user within 10 minutes, and can handle that a user burst call the getNextEvent API until queue is empty.

	Quota limits are set to protect OpenNet platform against excessive calling resulting in more http 204 responses than necessary.
Timeslot management API's	Users can do read modify write on all 365 possible future dates with one search and one update request per date. Users should not update data on a date unless it is actual needed, i.e. read and skip update unless data needs changing. As this is technicians doing manual labour, there is no need to update a full years timeslot every 30 minuttes. Users are encouraged to work smart, and limits will be set to alow a limited amount of full (365 dates) updates within a period of 24 hours. If users have different needs, please reach out to OpenNet and discuss further.

Table 44: Use of polling API's

In order to communicate current limits for each endpoint, ratelimit headers are returned in all API responses. All ratelimits are set as a number of requests to an endpoint name and version per minute.

An API endpoint is defined as URL/endpoint name/version. As an example enpoint name is "basket", "troubleticket" etc..

Header name	**Description**
X-RateLimit-Limit	Limit for API endpoint
X-RateLimit-Remaining	Number of requests remaining within window (1 minut)
X-RateLimit-Reset	Number of seconds remaining of current window (1 minut)

If the ratelimit is breached an http 429 – Too Many Requests is returned.

18. sFTP access

18.1 Pre-Production

Hostname: test.opennet.nu
Port number: 5222

18.2 Production

Hostname: prod.opennet.nu
Port number: 5222

19. Entity naming table

Entity Name	Service Provider Name	Service Provider Code	Additional Information
SP01	Nuuday	20000001	
SP02	OpenNet	20001001	OpenNet Test Service Provider
SP03	Hiper	20002001	
SP04	Altibox Danmark	20003001	

SP05	Eniig SP	20004001	
SP06	Telenor	20005001	
SP07	Kviknet	20006001	
SP08	Fibia	20007001	
SP09	Norlys Digital	20008001	
SP10	BoligNet	20009001	
SP11	Fastspeed	20010001	

Table 45: SP Entities