Date
2025-04-22

Supplement to industry recommendation

Conditional grid connections

Introduction

In addition to the recommendation published in Q4 of 2023, the DSOs saw a need to align on the details of OpenADR usage within the scope of conditional connection agreements communication. This includes aspect such as versioning, naming, formatting, formalized testing etc. In Q1 of 2024, an ad-hoc working group with DSO and CPO (Charge Point Operator, a class of grid customer often suited for conditional connection agreements) representation was formed to discuss and find common ground around these topics. This document collects findings from this work and may be used both to speed up the implementation of OpenADR, as well as to achieve a higher level of standardization among actors in the energy industry. It is mostly geared towards the DSO actors' possible needs of OpenADR VTN functionality in their day-to-day operation.

Similar to the recommendation, this addendum document does not go into too much detail since implementation of OpenADR requires following the official specification, which will likely change over time with version updates.

Contents

Energiföretagen Sverige
101 53 Stockholm, Besöksadress: Olof Palmes Gata 31
Tel: 08-677 25 00, E-post: info@energiforetagen.se, www.energiforetagen.se
Org. nr: 802000-7590, Säte: Stockholm

OpenADR comparison

As suggested in the recommendation, users of OpenADR should focus on the newest version of the specification. At the time of writing, that's the 3.0.1 version found on the OpenADR alliance web site.

To illustrate the main differences between major versions, below is a table listing some key properties:

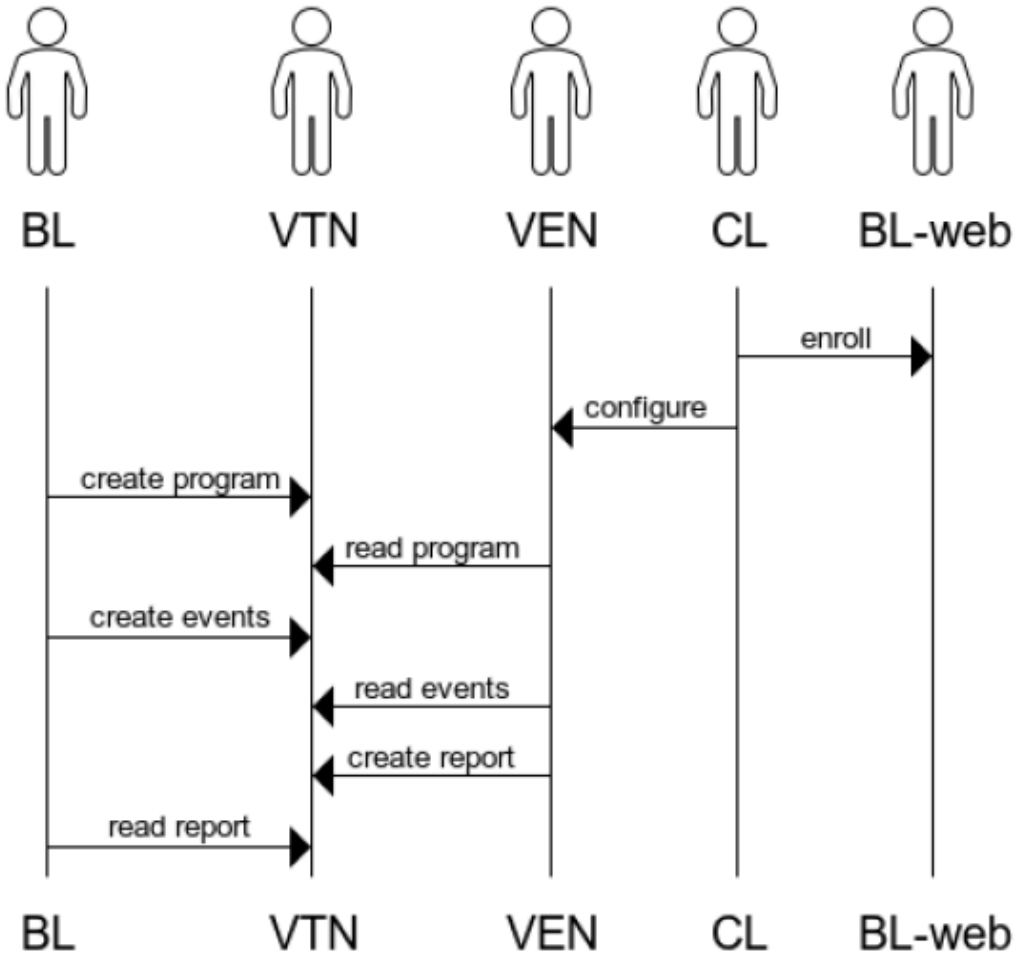
	2.0b	3.0
API format	• SOAP with HTTP/XMPP – Alliance profile specification – XML files	• RESTful based on OpenAPI standard – Official Swagger specification – JSON files, standard HTTP
Flows	• Pull or push – Clients can poll with HTTP – Server push with HTTP/XMPP	• Pull or push – Clients can poll with HTTP – Webhooks for push
Intended systems architecture	• Single VTN with multiple VEN – Strict hierarchy	• Multiple configurations possible – Business logic (DSO/server) – Customer logic (CPO/client) – BL/CL connects to VTN/VEN
Security model	• SSL/TLS client certificate – Requires customer to source certificate	• Provided by business logic – Common methods like Oauth2 can be used – Credentials managed outside VTN/VEN scope

One overall characteristic of 2.0b is that it's stricter than 3.0, which has both drawbacks and benefits. For instance, 2.0b relies on centrally authorized client certificates that can be cumbersome to configure and maintain, while 3.0 can be adapted to whatever security level is suitable for the application. On the other hand, the more predictable user flows can make initial registrations to 2.0b more streamlined than might be the case for a 3.0 implementation.

Due to the looser specification of 3.0, it's even more essential to find a common ground between DSOs especially when implementing the communication flows for conditional agreements. Another thing to consider carefully is the separation of business logic (e.g. access rights, authorization, event triggers, delivery validation etc.) from the core VTN/VEN, so that the application adheres to the OpenADR specification with minimal customization. Most of the infrastructure needed will be on the DSO side with whatever business logic is suitable for operational purposes, paired with a VTN that's as close as possible to a generalized solution in Sweden.

Information flows

OpenADR 3 provides several possibilities to facilitate the communication between the DSO actor (providing the VTN server) and the CPO (the connecting customer using a VEN client). On a high level, the official OpenADR 3 specification suggests the following setup:



- Business Logic (BL): a software client of a VTN that performs operations on behalf of an energy or energy service provider (i.e. the DSO).
- BL-web: a site or service for connecting customers, e.g. enrollment into an OpenADR3 program.
- VTN: OpenADR 3 resource server.
- VEN Client: a software client of a VTN that performs operations as requested by Business Logic via a VTN.
- Customer (not illustrated): a person that performs setup tasks prior to a VEN conducting interactions with a VTN.
- Customer Logic (CL): a software entity that embodies the functional logic within a VEN. A VEN is the software that communicates via the OpenADR 3 protocol, and the CL is the software, embodied within the VEN or separated via a VEN specific interface, that in turn interfaces with a set of energy resources.

- Resource (not illustrated): A device or system that performs operations as informed by program and events obtained by a VEN and provides data for VEN reports.

Within the scope of the industry recommendation, all actors wanting to implement OpenADR3 should adhere to this structure. Note that the VEN client can get information about events, reports etc. in multiple ways, e.g. by polling or webhooks. Implementing webhooks support in a VTN is recommended due to reduced overhead but requires additional effort to attain sufficient levels of security and robustness.

Proposed program parameters

OpenADR 3 introduces the program concept, as a way for VTNs to easily communicate their metadata to VENs. A VTN can have multiple programs, but generally one should suffice if the use case is common (e.g. conditional connections). The official specification contains an example of the information handled by programs:

program: Provides program specific metadata from VTN to VEN.
Id: VTN provisioned ID of this object instance.
createdDateTime: Creation time for object, e.g. "01012022:010000".
modificationDateTime: Modification time for object, e.g. "01012022:010000".
objectType: Used as discriminator. PROGRAM
programName: Name of program with which this event is associated, e.g. "ResTOU".
programLongName: User provided ID, e.g. "Residential Time of Use-A".
retailerName: Program defined ID, e.g. "ACME".
retailerLongName: Program defined ID, e.g. "ACME Electric Inc.".
programType: User defined string categorizing the program, e.g. "PRICING_TARIFF".
country: Alpha-2 code per ISO 3166-1, e.g. "US".
principalSubdivision: Coding per ISO 3166-2. E.g. state in US, e.g. "CO".
timeZoneOffset: Number of hours different from UTC for the standard time applicable to the program, e.g. "PT7H". [null]
intervalPeriod: The temporal span of the program, could be years long.
programDescriptions: List of URLs to human and/or machine-readable content, e.g. "mple: www.myCorporation.com/myProgramDescription".
bindingEvents: True if events are fixed once transmitted. [false]
localPrice: True if events have been adapted from a grid event. [false]
payloadDescriptors: An optional list of objects that provide context to payload types.
targets: An optional list of target objects.

Most of the listed parameters are not mandatory and can be ignored, since they wouldn't impact standardization efforts. A minimal version of a program could look like this:

```
{
    "id": "1",
    "createdDateTime": "2025-01-01T00:00:00.000Z",
    "modificationDateTime": "2025-01-01T00:00:00.000Z ",
    "objectType": "PROGRAM",
    "programName": "Conditional agreements",
    "timeZoneOffset": "PT0S",
    "programDescriptions": [],
    "bindingEvents": false,
    "localPrice": false,
    "payloadDescriptors": [],
    "targets": []
}
```

The "payloadDescriptors" parameter can be used to specify default event properties, but it's also possible to assign them for each event.

Proposed control event parameters

The most important entity for OpenADR communication is the event, which contains the actual information and payload for when and how to control a resource. There should be minimal variation between VTN events created for conditional agreements, and they should be as clearly structured as possible to simplify the communication. Like programs, the OpenADR specification provides an example of event information:

event: Event object to communicate a Demand Response request to VEN.
id: VTN provisioned ID of this object instance.
createdDateTime: server provisions timestamp on object creation, e.g. "01012022:010000".
modificationDateTime: server provisions timestamp on object modification, e.g. "01012022:010000".
objectType: Used as discriminator. EVENT
programID: ID attribute of program object this event is associated with.
eventName: User defined string for use in debugging or UI, e.g. "price event 11-18-2022".
priority: relative priority of event. A lower number is a higher priority.
targets: An array of target objects.
reportDescriptors: An array of reportDescriptor objects. Used to request reports from VEN.
payloadDescriptors: An array of payloadDescriptor objects.
intervalPeriod: Defines default start and durations of intervals.
intervals: An array of interval objects

Most of these parameters are already predefined by the program or the VTN (e.g. ID and created/modified timestamps). Targets should contain at least one identifier for the object of the event (i.e. a site, resource or asset depending on terminology). This may require business logic to manage multiple targets with different triggers, properties, rules etc. based on the customer agreements.

The generalized use case for automating conditional agreements could be described as follows:

1. A grid problem is detected by the DSO system
2. An event is created by the BL and VTN to limit one or more resources for a specified duration
3. The VEN sends a report to the VTN to acknowledge the limit
4. Additional events are created while the grid problem persists

In practical implementations of OpenADR3 for conditional connection agreements, two variants of event handling have been identified thus far. The first one (used by E.ON) has the following properties:

- Start time is always the next hourly aligned 15-minute period (e.g. 13:00, 13:15, 13:30 etc.) and duration is always “PT15M” (Period Time 15 Minutes, according to the ISO 8601 standard)
- Payload type is either “CONSUMPTION_POWER_LIMIT” or “PRODUCTION_POWER_LIMIT”, depending on what should be limited
- Unit of the limit is always “KW”, with the payload value indicating maximum allowed power during the event period
- While the grid issue persists, additional limit events may be sent if VEN resources need to be limited

- A report with payload type “POWER_LIMIT_ACKNOWLEDGEMENT” is requested for each event, to be sent immediately by the VEN

The second variant (used by Ellevio) has the following properties:

- Start time is real-time and immediate, represented as “0000-00-00T00:00:00.000Z”, and duration is typically “PT20M” (Period Time 20 Minutes, according to the ISO 8601 standard)
- Payload type is SIMPLE, with the payload value indicating either “Curtail” or “Restore”
- The curtailment limit is known by the VEN and is not included in the event payload
- 15 minutes after an event with the “Curtail” value is sent, a “Restore” event is sent
- If the grid issue is not resolved, an additional “Curtail” event is sent after 15 minutes
- A report with payload type “SIMPLE” is requested for each event, to be sent immediately by the VEN with a value of either “Executed” or “Not executed”

One important difference between the variants is that the first one requires the VEN resources to report their real-time power meter readings, while the second one does not. Depending on the DSO needs for delivery validation and dynamic limiting (i.e. the quantity of the grid breach can be distributed to multiple resources based on potential contributions), both variants could support the generalized use case.

Monitoring event and reporting

To decrease the risk of a communication issue leading to e.g. non-compliance of CPOs, additional steps for monitoring connection status may be introduced. For instance, the VTN could send regular “heartbeat” events to VENs, which would in turn respond with reports to indicate that their resources are online. Suggestions on payload structure can be found in the appendix section below.

Further standardization

Since the DSO requirements from e.g. commercial or operational perspective could vary, it’s challenging to find common ground beyond the above proposals. Regardless, more topics of standardization could be explored going forward:

- Additional use cases – events could be added or adapted to support other demand response scenarios, such as increasing production power by sending a threshold value.
- Reports for real-time CPO data – depending on how the DSO intends to validate the delivery compliance of the customer, meter data (e.g. measured active power) from the target object may need to be sent as reports to the VTN.
- Open source (e.g. client/server SDK, webhooks) – making tools, documentation and software publicly available should accelerate the adoption of OpenADR by lowering the threshold of entry for both DSOs and customers.

Additional resources

More information about the E.ON platform solution can be found here.

For more information about the Ellevio solution, please contact:

Emil Mossberg, Solution Architect, emil.mossberg@ellevio.se

Anders Ekberg, Vice President Operation & Maintenance, anders.ekberg@ellevio.se

Appendix: heartbeat payload examples

The suggested structure can be used with any reporting frequency that the DSO requires. The VEN is expected to send a report back for each event within a time period defined by the DSO. To simplify data extraction for the VEN, it's recommended to use a separate program for the heartbeat events. This also enables regular pruning of historic heartbeat events based on the data retention needs of the VTN.

Examples are given below of events both for targeting specific resources with IDs, as well as all resources on a VEN level using the generic "VEN_REPORT" value, with corresponding report samples. Payloads are kept minimal and follows recommendations of the OpenADR specification.

Heartbeat event sample targeting a specified resource:


```
{
 "id": "1122334455",
 "createdDateTime": "2025-03-13T13:43:00Z",
 "modificationDateTime": "2025-03-13T13:43:00Z",
 "objectType": "EVENT",
 "programID": "0",
 "eventName": "Heartbeat event 1122334455 modified 2025-03-13T13:43:00Z",
 "targets": [
  {
   "type": "RESOURCE_NAME",
   "values": ["2000"]
  }
 ],
 "intervalPeriod": {
  "start": "0000-00-00",
  "duration": "PT0S"
 },
 "reportDescriptors": [
  {
   "payloadType": "HEARTBEAT"
  }
 ],
 "intervals": [{
  "id": 0,
  "payloads": [{
   "type": "REQUIRED_BUT_NOT_USED",
   "values": [0]
  }]
 }]
}
```

Heartbeat event sample targeting a VEN or resource aggregation:


```
{
  "id": "1122334455",
  "createdDateTime": "2025-03-13T13:43:00Z",
  "modificationDateTime": "2025-03-13T13:43:00Z",
  "objectType": "EVENT",
  "programID": "0",
  "eventName": "Heartbeat event 1122334455 modified 2025-03-13T13:43:00Z",
  "intervalPeriod": {
    "start": "0000-00-00",
    "duration": "PT0S"
  },
  "targets": [
    {
      "type": "RESOURCE_NAME",
      "values": ["VEN_REPORT"]
    }
  ],
  "reportDescriptors": [
    {
      "payloadType": "HEARTBEAT"
    }
  ],
  "intervals": [{
    "id": 0,
    "payloads": [{
      "type": "REQUIRED_BUT_NOT_USED",
      "values": [0]
    }]
  }]
}
```

Heartbeat report sample from a specified resource:


```
{
  "id": "3000",
  "createdDateTime": "2025-03-13T13:44:00Z",
  "modificationDateTime": "2025-03-13T13:44:00Z",
  "objectType": "REPORT",
  "programID": "0",
  "eventID": "1122334455",
  "clientName": "1234567",
  "reportName": "Heartbeat event 1122334455 executed 2025-03-13T13:44:00Z",
  "payloadDescriptors": [
   {
    "payloadType": "HEARTBEAT"
   }
  ],
  "resources": [
   {
    "resourceName": "2000",
    "intervalPeriod": {
     "start": "2025-03-13T13:44:00Z",
     "duration": "PT0S"
    },
    "intervals": [
     {
      "id": 0,
      "payloads": [
       {
        "type": "HEARTBEAT",
        "values": ["OK"] // or NOT_OK
       }
      ]
     }
    ]
   }
  ]
}
```

Heartbeat report sample from a VEN or resource aggregation:


```
{
  "id": "3000",
  "createdDateTime": "2025-03-13T13:44:00Z",
  "modificationDateTime": "2025-03-13T13:44:00Z",
  "objectType": "REPORT",
  "programID": "0",
  "eventID": "1122334455",
  "clientName": "1234567",
  "reportName": "Heartbeat event 1122334455 executed 2025-03-13T13:44:00Z",
  "payloadDescriptors": [
   {
    "payloadType": "HEARTBEAT"
   }
  ],
  "resources": [
   {
    "resourceName": "VEN_REPORT",
    "intervalPeriod": {
     "start": "2025-03-13T13:44:00Z",
     "duration": "PT0S"
    },
    "intervals": [
     {
      "id": 0,
      "payloads": [
       {
        "type": "HEARTBEAT",
        "values": ["OK"] // or NOT_OK
       }
      ]
     }
    ]
   }
  ]
}
```