



BSI Standards Publication

Systems interface between customer energy management system and the power management system

Part 10-1: Open automated demand response

National foreword

This British Standard is the UK implementation of IEC 62746-10-1:2018.

The UK participation in its preparation was entrusted to Technical Committee L/13, Smart Energy Systems coordination group.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2019
Published by BSI Standards Limited 2019

ISBN 978 0 580 89672 9

ICS 33.200

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 August 2019.

Amendments/corrigenda issued since publication

Date	Text affected
------	---------------

INTERNATIONAL STANDARD



**Systems interface between customer energy management system and the power management system –
Part 10-1: Open automated demand response**



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2018 IEC, Geneva, Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester. If you have any questions about IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

IEC Central Office
3, rue de Varembe
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

IEC Catalogue - webstore.iec.ch/catalogue

The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

IEC publications search - webstore.iec.ch/advsearchform

The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,...). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and also once a month by email.

Electropedia - www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing 21 000 terms and definitions in English and French, with equivalent terms in 16 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary

67 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IEC Customer Service Centre - webstore.iec.ch/csc

If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

CONTENTS

FOREWORD.....	5
INTRODUCTION.....	7
1 Scope.....	9
2 Normative references	9
3 Terms, definitions and abbreviated terms	10
3.1 Terms and definitions.....	10
3.2 Abbreviated terms.....	11
4 Overview	12
4.1 General.....	12
4.2 Node and device types.....	13
4.3 IEC 62746-10-1 services.....	14
4.4 Assumptions	15
5 IEC 62746-10-1 features.....	15
5.1 General.....	15
5.2 Supported services	15
5.3 Report Only VENs.....	15
5.4 Transport mechanism	15
5.5 Security	15
6 Services and data model extensions.....	15
6.1 Event service	15
6.1.1 Event interactions.....	15
6.1.2 oadrEvent mechanism	19
6.2 Report service	23
6.2.1 General	23
6.2.2 Core reporting operations	25
6.3 Registration service	30
6.3.1 Service operations.....	30
6.3.2 Registration information.....	33
6.4 Opt service	34
6.4.1 Service operations.....	34
6.4.2 Detail requirements	35
6.5 Poll service	36
6.6 Application error codes	39
7 Transport protocol	40
7.1 General.....	40
7.2 Simple HTTP	40
7.2.1 General	40
7.2.2 PUSH and PULL implementation	40
7.2.3 Service endpoint URIs	41
7.2.4 HTTP methods.....	41
7.2.5 Failure conditions	41
7.2.6 HTTP response codes	41
7.2.7 Message timeouts.....	42
7.2.8 Message retry/quiesce behaviour	42
7.2.9 PULL timing.....	42
7.2.10 HTTP headers	42

7.3	Transport-specific security	44
7.3.1	General	44
7.3.2	TLS client certificate	44
7.4	XMPP	44
7.4.1	General	44
7.4.2	Exchange model implementation	44
7.4.3	Service endpoints	44
7.4.4	Service execution	45
7.4.5	Implementation of XMPP features	45
7.4.6	Security considerations	48
8	Cyber security	49
8.1	General	49
8.2	Architecture and certificate types	49
8.3	Certificate authorities	50
8.4	Certificate revocation	50
8.5	TLS and cipher suites	50
8.6	System registration process	50
8.6.1	General	50
8.6.2	Certificate fingerprints	50
8.7	Implementing XML signatures for message payloads	51
8.7.1	XML signature	51
8.7.2	Components of XML signatures	51
8.7.3	Creating XML signatures	52
8.7.4	Verifying XML signatures	53
9	Conformance	53
9.1	Conformance statement	53
9.2	Conformance rules	53
9.2.1	EiEvent	53
9.2.2	EiEvent – Additional 2.0b conformance rules	61
9.2.3	EiOpt	63
9.2.4	EiReport	65
9.2.5	EiRegisterParty	71
9.2.6	General conformance rules	73
9.3	Cardinality	76
Annex A (normative)	Detailed report description	77
Annex B (normative)	Profile extensions	78
B.1	Overview	78
B.2	Report extension	78
B.3	Event extension	78
B.4	Other extensions	78
Annex C (normative)	oadrPoll scenarios	79
C.1	Overview	79
C.2	Scenarios	79
Annex D (normative)	Definition of VEN, VTN, resource, and party	81
Annex E (normative)	IEC 62746-10-1 Schema	82
Bibliography		206

Figure 1 – Possible relationships of VTN and VEN.....

Figure 2 – EiEvent PUSH pattern	16
Figure 3 – EiEvent PULL pattern	17
Figure 4 – Time intervals of an event	19
Figure 5 – Report type	24
Figure 6 – Interaction diagram: Register reporting capabilities	26
Figure 7 – Interaction diagram: Request reports	27
Figure 8 – Interaction diagram: Send reports	29
Figure 9 – Interaction diagram: Cancel reports	30
Figure 10 – Interaction diagram: Query registration	31
Figure 11 – Interaction diagram: Create registration	31
Figure 12 – Interaction diagram: Request reregistration	32
Figure 13 – Interaction diagram: Cancel registration	32
Figure 14 – Interaction diagram: Create opt	35
Figure 15 – Interaction diagram: Cancel opt	35
Figure 16 – Interaction diagram: oadrPoll (nothing in queue)	37
Figure 17 – Interaction diagram: oadrPoll (oadrDistributeEvent reply)	38
Figure 18 – Interaction diagram: oadrPoll (oadrCreateReport reply)	38
Figure 19 – Interaction diagram: oadrPoll (request reregistration reply)	39
Figure 20 – XML signature example	52
Table 1 – IEC 62746-10-1 services support	13
Table 2 – Signals	21
Table 3 – EiRegisterParty payloads	30
Table 4 – VEN information in oadrCreatePartyRegistration payload	33
Table 5 – VTN information oadrCreatedPartyRegistration payload	34
Table 6 – EiOpt payloads	34
Table 7 – Conformance rules	54
Table 8 – Additional conformance rules	61
Table 9 – EiOpt conformance rules	63
Table 10 – EiReport conformance rules	65
Table 11 – EiRegisterParty conformance rules	71
Table 12 – General conformance rules	73
Table 13 – Cardinalities	76

INTERNATIONAL ELECTROTECHNICAL COMMISSION

**SYSTEMS INTERFACE BETWEEN CUSTOMER ENERGY
MANAGEMENT SYSTEM AND THE POWER MANAGEMENT SYSTEM –****Part 10-1: Open automated demand response**

FOREWORD

- 1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as “IEC Publication(s)”). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
- 2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
- 3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
- 5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
- 6) All users should ensure that they have the latest edition of this publication.
- 7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
- 8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
- 9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 62746-10-1 has been prepared by IEC project committee 118: Smart grid user interface.

The text of this International Standard is based on the following documents:

CDV	Report on voting
118/91/CDV	118/96B/RVC

Full information on the voting for the approval of this International Standard can be found in the report on voting indicated in the above table.

This document has been drafted in accordance with the ISO/IEC Directives, Part 2.

A list of all parts in the IEC 62746 series, published under the general title *Systems interface between customer energy management system and the power management system*, can be found on the IEC website.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under "<http://webstore.iec.ch>" in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

A bilingual version of this publication may be issued at a later date.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

Development of the demand response (DR) market has resulted in a transition from manual DR to OpenADR (this document) in automated DR programs. DR is defined as an action taken to reduce electricity demand in response to price, monetary incentives, or utility directives so as to maintain reliable electric service or avoid high electricity prices. This document was developed to support common auto-DR programs and energy policy objectives to move toward dynamic markets to improve the economics and reliability of the electricity grid. The recent developments have expanded the use of this document to meet diverse market needs, such as ancillary services, dynamic prices, intermittent renewable resources, supplement grid-scale storage, electric vehicles, and load as generation. For example, with real-time price information, an automated client within the customer facility can be designed to continuously monitor these prices and translate this information into continuous automated control and response strategies.

This document's communication has the following defining features:

- Continuous, secure, and reliable – Provides continuous, secure, and reliable two-way communications where the endpoints at the end-use site receive and acknowledge the receipt of DR signals from the energy service providers.
- Translation – Translates DR event information to internet signals to facilitate DR automation. These signals are designed to interoperate with energy management and control systems, lighting, or other end-use controls.
- Automation – Receipt of the external signal is designed to initiate automation through the use of pre-programmed demand response strategies determined and controlled by the end-use participant.
- Opt-out – Provides opt-out or override function to any participants for a DR event if the event comes at a time when changes in end-use services are not desirable.
- Complete data model – Describes a rich data model and architecture to communicate price, reliability, and other DR activation signals.
- Scalable architecture – Provides scalable communications architecture to different forms of DR programs, end-use buildings, and dynamic pricing.
- Open standards – Open standards-based technology such as internet protocol (IP) and web services form the basis of the communications model.

This document is a communications data model, along with transport and security mechanisms, which facilitate information exchange between two end-points: the electricity service provider or DR program operator, and a customer-side resource. It is not a protocol that specifies "bit-structures" as some communications protocols do, but instead relies upon existing open standards such as Extensible Markup Language (XML) and internet protocol (IP) as the framework for exchanging DR signals. In some references, the term "system," "technology," or "service" is used to refer to the features of this document.

IEC 62746-10-1 is designed to facilitate automation of DR actions at the customer location, whether it involves electric load decreases, load increases, or load shifting for various demand response markets. Many emergency or reliability DR events occur at specific times when the electricity grid is strained. The communications are designed to coordinate such signals with facility control systems (commercial, industrial, and residential). This document is also designed to provide continuous dynamic price signals, such as hourly, day-ahead, or day-of real-time pricing. With such price information, an automated client can be configured to continuously monitor these prices and translate this information into continuous automated control and response within a facility. Several reports present the history of this document and the involved research. This document covers the signalling data models for price and reliability signals to both wholesale and retail markets, and can act as a complementary standard to a CIM-based grid control system.

This document provides the following benefits:

- Open specification – Provides a standardized DR communications and signalling infrastructure using open, non-proprietary, industry-approved data models that can be implemented for both dynamic prices and DR emergency or reliability events.
- Flexibility – Provides open communications interfaces and protocols that are flexible, platform-independent, interoperable, and transparent to end-to-end technologies and software systems.
- Innovation and interoperability – Encourages open innovation and interoperability, and allows controls and communications within a facility or enterprise to build on existing strategies to reduce technology operation and maintenance costs, stranded assets, and obsolescence in technology.
- Ease of integration – Facilitates integration of common energy management and control systems (EMCS), centralised lighting, and other end-use devices that can receive internet signals (such as XML).
- Supports a wide range of information complexity – Can express the information in the DR signals in a variety of ways to allow for systems ranging from simple end devices (e.g. thermostats) to sophisticated intermediaries (e.g. aggregators) to receive the DR information that is best suited for its operations.

This document's purpose is to manage the growing demand for demand-side flexibility – the ability to modify the load profile of the consumers over time. This low-cost communications infrastructure is used to improve the reliability, repeatability, robustness, and cost-effectiveness of DR.

This technology has been field tested for over a decade and deployed in a number of DR programs worldwide. While the scope of this document focuses on signals for DR events and prices, significant supporting research work has been done to study DR strategies and techniques to automate DR within facilities. This document interacts with facility control systems that are pre-programmed to take action based on a DR signal, enabling a response to a DR event or a price to be fully automated, with no manual intervention.

SYSTEMS INTERFACE BETWEEN CUSTOMER ENERGY MANAGEMENT SYSTEM AND THE POWER MANAGEMENT SYSTEM –

Part 10-1: Open automated demand response

1 Scope

This part of IEC 62746-10, OpenADR 2.0 (this document), specifies a minimal data model and services for demand response (DR), pricing, and distributed energy resource (DER) communications. This document can be leveraged to manage customer energy resources, including load, generation, and storage, via signals provided by grid and/or market operators. These resources can be identified and managed as individual resources with specific capabilities, or as virtual resources with an aggregated set of capabilities.

This document specifies how to implement a two-way signaling system to facilitate information exchange between electricity service providers, aggregators, and end users. The DR signalling system is described in terms of servers (virtual top nodes or VTNs), which publish information to automated clients (virtual end nodes, or VENs), which in turn subscribe to the information.

This document provides application-level service communication that can be used to incentivise a response from a customer-owned DER. Price and DR signals over the internet allow indirect control of customer-owned devices that otherwise would not be available.

This document's services are independent of transport mechanisms. For the purposes of interoperability, this document provides basic transport mechanisms and their relevant interaction patterns to address different stakeholder needs. In addition, this document specifies cyber security mechanisms required for data confidentiality, integrity, authentication and message-level security, in order to provide non-repudiation and mitigation of cyber security risks.

The services make no assumption of specific DR electric load control strategies that can be used within a DR resource or of any market-specific contractual or business agreements between electricity service providers and their customers.

This document provides a clear set of mandatory and optional attributes within each of the services to meet broader interoperability, testing and certification requirements. All necessary XML schema are included in Annex E.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 8601, *Data elements and interchange formats – Information interchange – Representation of dates and times*

INTERNET ENGINEERING TASK FORCE (IETF). RFC 2616: *Hypertext Transfer Protocol – HTTP/1.1* [online]. Edited by R. Fielding et al. June 1999 [viewed 2018-08-02]. available at: <http://www.ietf.org/rfc/rfc2616.txt>

INTERNET ENGINEERING TASK FORCE (IETF). RFC 3986: *Uniform Resource Identifier (URI): Generic Syntax* [online]. Edited by T. Berners-Lee et al. January 2005 [viewed 2018-08-02]. available at:
<http://www.ietf.org/rfc/rfc3986.txt>

INTERNET ENGINEERING TASK FORCE (IETF). RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2 [online]. Edited by T. Dierks et al. August 2008 [viewed 2018-08-02]. available at
<https://tools.ietf.org/html/rfc5246>

INTERNET ENGINEERING TASK FORCE (IETF). RFC 6120: *Extensible Messaging and Presence Protocol (XMPP): Core* [online]. Edited by P. Saint-Andre. March 2011 [viewed 2018-08-02]. available at:
<http://www.ietf.org/rfc/rfc6120.txt>

INTERNET ENGINEERING TASK FORCE (IETF). RFC 6121: *Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence* [online]. Edited by P. Saint-Andre. March 2011 [viewed 2018-08-02]. available at:
<http://www.ietf.org/rfc/rfc6121.txt>

XMPP Standards Foundation. *XEP-0030: Service Discovery* [online]. Edited by J. Hildebrand et al. October 2017 [viewed 2018-08-02]. available at:
<http://xmpp.org/extensions/xep-0030.html>

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at <http://www.electropedia.org/>
- ISO Online browsing platform: available at <http://www.iso.org/obp>

3.1.1

OpenADR 2.0

specification that provides the basis for this document

3.1.2

PUSH and PULL operations

mode of operation by which this document can be used in either PULL mode (VEN pulling information from VTN) or in a PUSH mode (VTN pushing information to the VEN)

Note 1 to entry: The XMPP transport uses a PUSH model, although VENs can still make requests of the VTN, excluding the use of oadrPoll.

3.1.3

simple HTTP

HTTP implementation that uses HTTP POST over TLS to propagate payloads

3.1.4

virtual end node

VEN

technical role assumed by an actor where the actor is a consumer and/or producer of messages that are defined by this document

Note 1 to entry: A virtual end node (VEN) can be associated with zero or more resources. A VEN can receive messages pushed from a VTN or send requests or events to a VTN. A VEN may communicate with multiple VTNs, where each VTN is part of a different communication domain.

[SOURCE: IEC TS 62746-3:2015, 3.1.15, modified – Note 2 to entry has been omitted.]

3.1.5

virtual top node

VTN

technical role assumed by an actor that is assuming responsibility for the coordination of VENs within a communication domain

Note 1 to entry: This is a special case of a VEN, where a virtual top node (VTN) is effectively a parent of many VENs with the responsibility for coordination of those VENs. A VTN is responsible for pushing to or receiving messages from many VENs. A market operator, grid operator or aggregator are examples of actors which will typically implement a VTN interface.

[SOURCE: IEC TS 62746-3:2015, 3.1.17, modified – Note 2 to entry has been omitted.]

3.1.6

resource

demand-side commodity that is associated with a load profile

3.1.7

party

entity that enters into some sort of business relationship or contract

Note 1 to entry: A more detailed definition can be found in Annex D.

3.1.8

RSA

public-key crypto system placed into the public domain by RSA Data Security, Inc.

3.2 Abbreviated terms

CA	certificate authority
DER	distributed energy resources
DR	Demand Response
ECC	elliptic curve cryptography
EI	energy interoperation
HTTP	Hypertext Transfer Protocol
ISO	independent systems operator
JID	jabber identifier
OpenADR	Open Automated Demand Response
PICS	protocol implementation conformance statement
PKI	public key infrastructure
SASL	Simple Authentication and Security Layer
SOAP	Simple Object Access Protocol
TLS	Transport Layer Security
UCAIug	Utilities Communications Architecture International Users Group
VEN	virtual end node
VTN	virtual top node
XML	Extensible Markup Language
XMPP	XML Messaging and Presence Protocol

4 Overview

4.1 General

Clause 4 gives an overview of the message exchanges, the roles, and actors supported within this document. It contains the following elements that are used to develop test and certification frameworks for smart grid and customer system interoperability:

- a) a set of data models that describe information communicated in message payloads;
- b) a set of services for performing various functions and operations for the exchange of the data models;
- c) a set of transport mechanisms for implementing the services. The transport mechanisms rely upon standard-based IP communications, such as HTTP and XML messaging and presence protocol (XMPP);
- d) a set of security mechanisms for securing each of the transport mechanisms;
- e) a set of XML schemas (see Annex E).

Integration of IEC 62746-10-1 systems within the IEC's standards framework is done with a CIM adapter that may be produced in accordance with the methodology described in IEC 62746-10-3. Message exchanges in this document support services for communicating information about demand response events. Networks of nodes shall be able to query for active or pending events, register themselves, schedule events, and send reports. The nodes shall also be able to refine and update previously sent information. For instance, a node reporting DR events to nodes downstream shall be able to cancel a previously scheduled event if this becomes necessary.

Nodes in these networks are divided into two groups: nodes that publish and transmit information about events to other nodes (e.g. utilities), and nodes that receive the communications and then respond to that information (e.g. end users). The upstream nodes that publish information about upcoming events are called virtual top nodes (VTNs); the downstream nodes that receive this information are called virtual end nodes (VENs).

These nodes may communicate using a variety of protocols. They may communicate using HTTP in either PUSH mode (where the VTN initiates communication) or in a PULL mode (the VEN requests information from the VTN to begin a series of message exchanges). The VTNs/VENs may also communicate over other transport mechanisms, such as XML messaging and presence protocol (XMPP).

This document supports end devices with a varying degree of capabilities. However, for interoperability, all protocol capabilities are mandatory.

IEC 62746-10-1 specifies the following services:

- 1) Register: Registration identifies entities in advance of interactions with other parties in various roles such as VEN and VTN.
- 2) Event: The core DR event functions and information models for price-responsive DR. This service is used to call for performance under a transaction. The service parameters and event information distinguish different types of events: reliability events, emergency events, price events, regulation events and possibly other types in the future.
- 3) Report: The report service enables feedback to the server in order to provide periodic or one-time information on the state of a resource.
- 4) Opt: Overrides the EiAvail and addresses short-term changes in availability to create and communicate opt-in and opt-out schedules from the VEN to the VTN.

Table 1 outlines the mandatory and optional implementation of this document.

Table 1 – IEC 62746-10-1 services support

	VTN	VEN	
		Full	(Energy Reporting only)
Services and Functions Support			
EiEvent			
Full Profile	M	M	NA
EiOpt			
Full Profile	M	M	NA
EiReport			
Full Profile	M	M*	M*
EiRegisterParty			
Full Profile	M	M	M
Transport Protocols			
Simple HTTP	M	O-1	O-1
XMPP	M	O-1	O-1
Security Levels			
Standard	M	M	M
High	O	O	O
M Mandatory O Optional O-1 Optional, but at least one option shall be supported NA Not available * Optional features available			

4.2 Node and device types

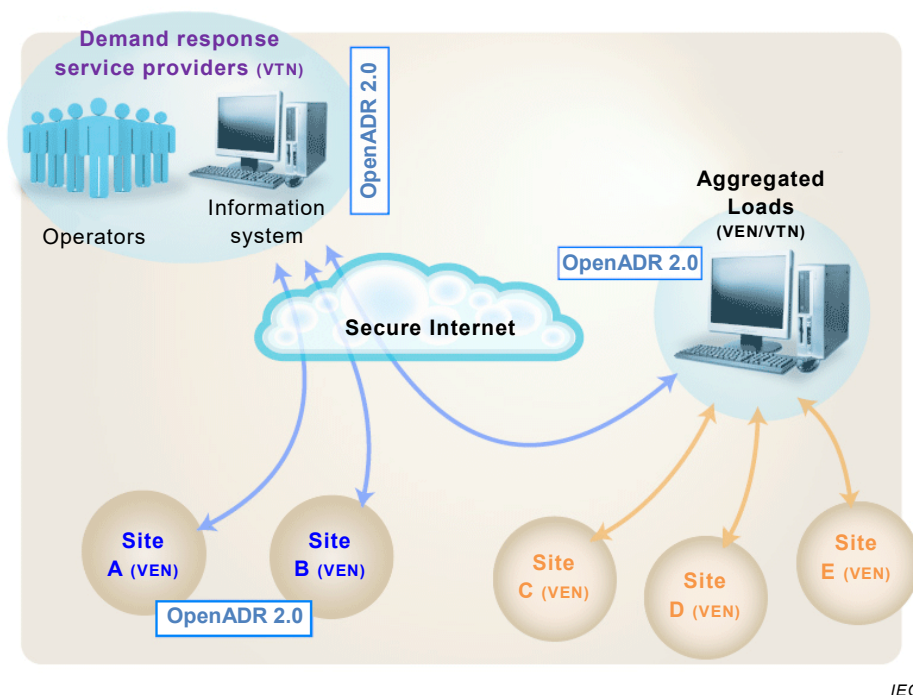
For any interaction between actors using this document to communicate, one actor is designated the virtual top node and the remainder are the virtual end nodes. All communications are between a VTN and one or more VENs. There is no peer-to-peer communication in this document, i.e. VTNs do not communicate directly with other VTNs, and, likewise, VENs do not communicate directly with other VENs.

The VTN generally acts as the server, providing information to the VEN, which responds to the information. For instance, a VTN would be the entity to announce a DR event; then, VENs hear about DR events and respond. The response can be to reduce power to some devices. The response could also be to propagate the signal further downstream to other VENs. In this case, the VEN would become the VTN for the new interaction (e.g. the aggregator in Figure 1).

For the purpose of device development, the manufacturer should always test the interface between a VTN and a VEN, where either node can be the device under test. Intelligence built into the systems not related to the message exchange is not part of this document.

The authoritative requirements for implementation of VENs and VTNs are defined in the schema and the conformance rules. Narrative descriptions in later clauses separate from the conformance rules provide context and implementation examples, but do not contain the full breadth of implementation requirements. Therefore, it is strongly recommended that implementers thoroughly understand all the conformance rules prior to coding.

Although within any interaction, one actor is designated the VTN and the remainder are VENs (moreover, most interactions have exactly one VTN and one VEN), sets of actors can be arranged in any hierarchy, by allowing actors to act as VENs for some interactions and VTNs for others.



IEC

Figure 1 – Possible relationships of VTN and VEN

As illustrated in Figure 1, any combination of VTN and VEN is possible through a utility/ISO (service provider or server) to sites (customers). Also, as shown above, systems can function as a VEN to a VTN in a higher layer of the hierarchy and as a VTN to subordinate VENs. In either of these architectural scenarios, an operation can be initiated by the VTN to a VEN (PUSH pattern) or a VEN can request it from a VTN (PULL pattern). The exchanged events in either direction can be independent from each other and this document does not define how the nodes react to the information. In nodes that support both the VTN and VEN interfaces (e.g. aggregators), there are no specifications or constraints on how messages arriving at the VEN interface are coupled or translated into any subsequent messages that can be sent from the VTN interface, and vice versa. They are treated as completely independent interfaces and both will be evaluated and tested independently to assure adherence to the profile specification and interoperability. A specific deployment scenario depends on an agreement between the utility/ISO and the participating sites.

4.3 IEC 62746-10-1 services

This document uses four defined services, each one profiled and tailored to meet the relevant market needs. These services are EiRegisterParty, EiEvent, EiReport, and EiOpt. For further information on these services, consult Clause 6.

4.4 Assumptions

Operating assumptions regarding correct functional behaviour between VENs and VTNs are as follows:

- Both the VTN and the VEN shall have awareness of the current time. Time synchronization requirements are determined by the specific implementation and demand response program.

5 IEC 62746-10-1 features

5.1 General

This document's feature set was developed for advanced demand response systems and markets (e.g. wholesale and retail DR markets). It includes an extended EiEvent service as well as several additional services usable in demand response programs and for ancillary services.

5.2 Supported services

- a) EiEvent service
- b) EiReport service
- c) EiRegisterParty service
- d) EiOpt service

5.3 Report Only VENs

Some devices, such as meters, do not have the ability to shed load. However, these types of devices can provide valuable reporting information to the VTN. This document has a sub-profile for VENs called Report Only, which includes support for the following services:

- a) EiReport service;
- b) EiRegisterParty Service.

5.4 Transport mechanism

Supported transport mechanisms are as follows. The mechanisms are described in Clause 7.

- a) HTTP is mandatory for VTNs; VENs shall either support HTTP or XMPP;
- b) XMPP is mandatory for VTNs; VENs shall either support HTTP or XMPP.

5.5 Security

Supported security details are outlined in Clauses 7 and 8. The following security levels apply to this document:

- a) standard security – mandatory;
- b) high security – optional.

6 Services and data model extensions

6.1 Event service

6.1.1 Event interactions

Events are generated by the VTN and sent to the VEN using the oadrDistributeEvent payload containing one or more events described by the oadrEvent element. Some events require a response and others do not, as indicated by the oadrResponseRequired element in the event description. If a response is required, the VEN acknowledges its opt-in or opt-out disposition by responding with an oadrCreatedEvent payload containing eventResponse elements.

matching each `oadrEvent`. If no response is required, the VEN shall not reply with `oadrCreatedEvents` (or `oadrCreateOpt`) payloads for this event.

Either a PUSH or PULL interaction pattern may be used. For push, the VTN shall deliver events to the VEN using an `oadrDistributeEvent` payload. In PULL mode, the `oadrDistributeEvent` shall be sent from the VTN to the VEN as response to an `oadrPoll` (refer to 6.5). In addition to periodically sending `oadrPoll`, a VEN may also send one-time `oadrRequestEvent` payloads to the VTN in order to acquire events from the VTN. If an application level response is required, the VEN asynchronously sends an `oadrCreatedEvent` back to the VTN in a second message. These sequences are illustrated in Figure 2. Note that in simpleHTTP PUSH mode, the VEN's response to `oadrDistributeEvent` is a transport level acknowledgement if required (in the case of HTTP, a 200 response code; in XMPP, an empty IQ stanza).

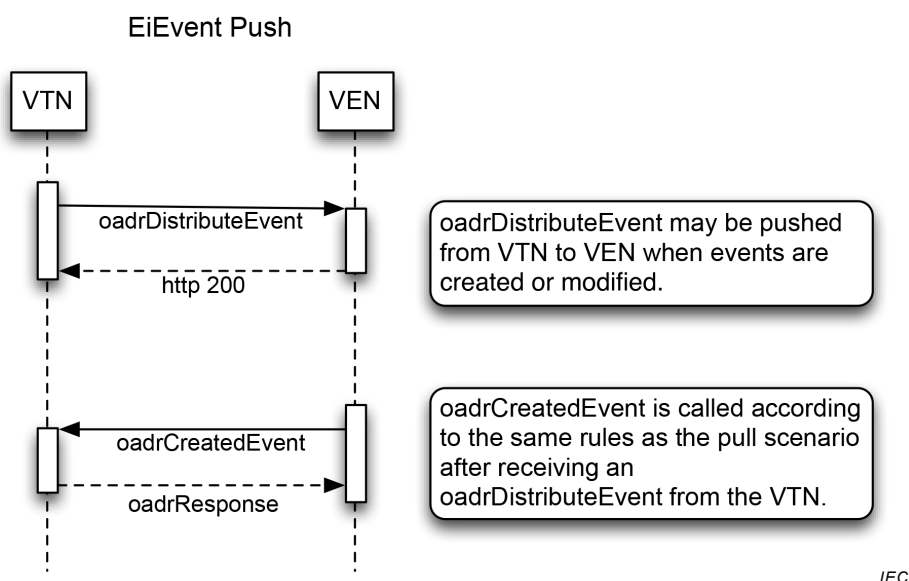


Figure 2 – EiEvent PUSH pattern

For the PULL case, the VEN requests events by sending an `oadrPoll` to the VTN (see 6.5). The VTN responds with an `oadrDistributeEvent`. From this point, the VEN response is exactly the same as in the PUSH interaction. These sequences are illustrated in Figure 3.

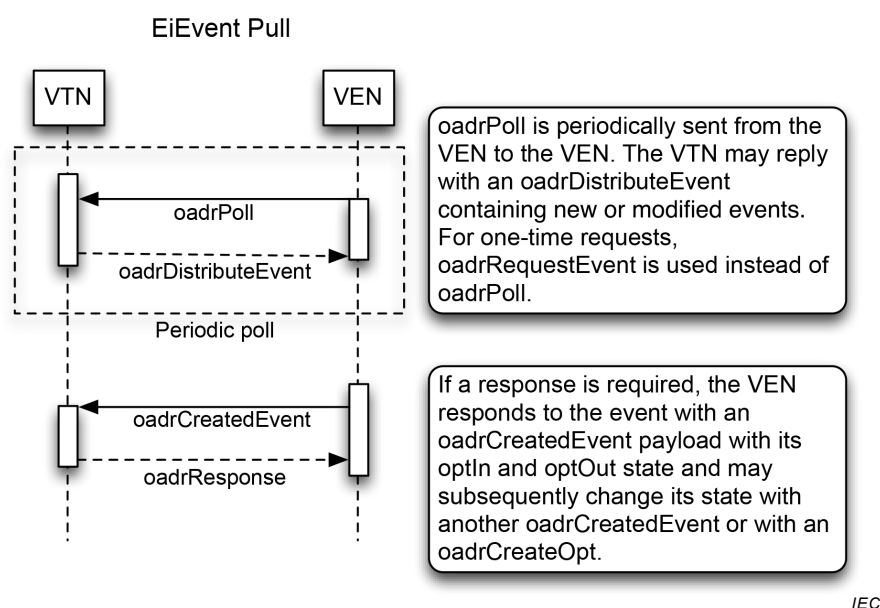


Figure 3 – EiEvent PULL pattern

The details of how these interactions are performed within the context of a specific transport mechanism are covered in Clause 7.

When an event requires a response, an initial `oadrCreatedEvent` is always sent from the VEN to the VTN. If a given program allows a VEN to later change its `opt`-state during an event, it may do so by issuing a subsequent `oadrCreatedEvent` (or `oadrCreateOpt`) message containing the new state for a given event. Detailed descriptions of VTN and VEN event processing are given in the following paragraphs.

Note that for both PUSH and PULL operations, an `oadrDistributeEvent` payload will always contain all events applicable to the VEN it is communicating with.

Events are conveyed in the `oadrDistributeEvent` payload using one or more `oadrEvent` elements. The `oadrDistributeEvent` payload has the following components:

- a `requestID` to uniquely identify this request and any contained events;
- a `vtnID` identifying the VTN sending the event;
- zero or more `oadrEvent` elements.

The `requestID` uniquely identifies the request and any contained events. Its value is set by the VTN using any scheme desired, including using the same value in every request if its use is not needed by the VTN. The receiving VEN shall use this `requestID` in the `oadrCreatedEvent` event responses.

a) `oadrEvent` description

`oadrEvent` elements describe individual events, signal values, and time periods that apply to signals. Each `oadrEvent` has an `eiEvent` element containing detailed event information and an `oadrResponseRequired` element that controls whether a VEN shall respond with an `oadrCreatedEvent`.

The `responseRequired` field indicates how the VEN shall respond to contained events. A value of "always" indicates that the VEN shall reply to each event whether or not the event state (`eventID`, `modificationNumber`) has changed. A value of "never" implies a "broadcast" event and the VEN shall not send any responses in this case.

The eiEvent eventDescriptor has the following fields:

- eventID – a unique ID for this event. The ID is unique within the context of a VEN.
- modificationNumber – A sequence that starts at zero and is incremented by 1 each time the VTN modifies the event.
- modificationDateTime – The time the event has been modified (only in 2.0b payloads). Refer to 5.4 for time synchronization requirements.
- modificationReason – The reason why the event has been modified (only in 2.0b payloads).
- priority – An indication of the event priority with 0 being no priority.
- marketContext – Identifies a particular program or application-defined grouping that pertains to an event.
- createdDateTime – The time the payload containing the event was created. Note that payloads are not necessarily created each time they are transmitted as they can be buffered and only recreated if the event is modified. Any change of the payload of the event requires an update of createdDateTime. Refer to 5.4 for time synchronization requirements.
- eventStatus – The status of the event, indicating if the event is "near", "far", "active" or "cancelled".
- testEvent – If not false, indicates this is a test event.
- vtnComment – Arbitrary comment provided by the VTN.

A single eiActivePeriod defines the start time and duration of the event. The start time is defined by an ISO 8601 time descriptor and an ISO 8601 duration string specifies the duration. Implementations are required to support the ISO 8601 representation (including support for fractional seconds).

Event start times can be randomized using the tolerance object in the eiActivePeriod. The sub-element startafter defines a randomization time window used by the VEN to select a random value that is added to the start time of the event. If the start time of a one-hour event is 3:00 and the randomization window is 5 min, if the VEN selected 3 min as the random value then the event would start at 3:03 and would end at 4:03.

The event signals that get applied over the entire active period are defined in an eiEventSignals element. This element contains one or more eiEventSignal elements, each with a sequence of durations, the sum of which shall equal the full duration of the active period. Each signal element contains a signalType, such as level or price. The signalPayload contains the value of the signal according to Table 2.

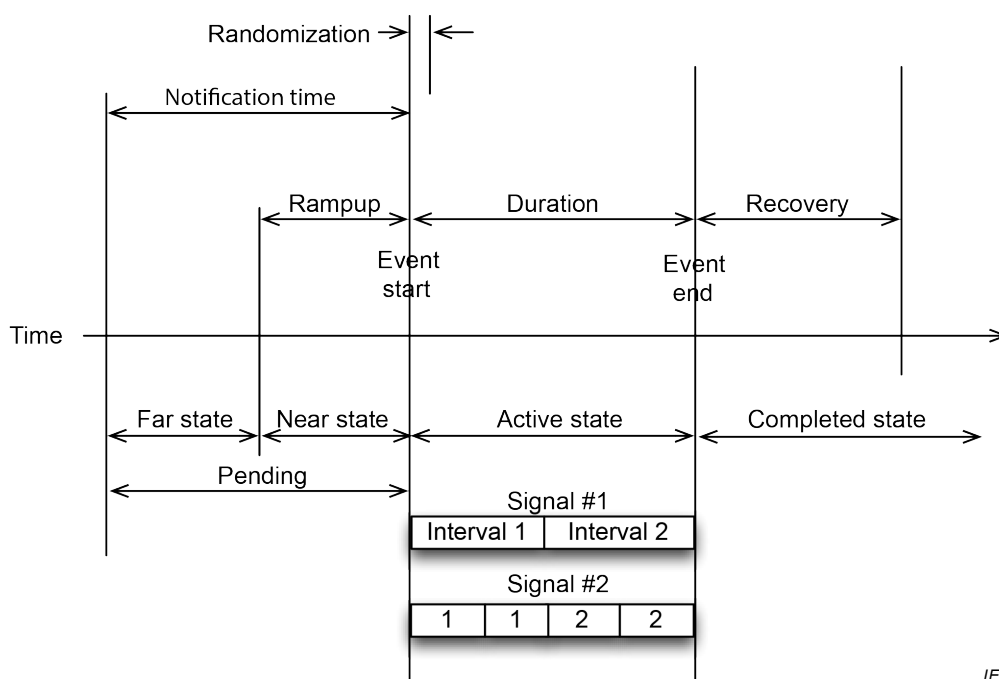
Figure 4 depicts the different time periods of an event. Two items on this figure not previously discussed are defined below.

Ramp Up Period:

Period at the beginning of the Active Interval expressed as a Duration, during which a VEN moves from its former state to its requested state. If negative, then the Ramp Up occurs within the bounds of the Active Interval, i.e. it starts at the same moment as the Active Interval. If there is no Ramp Up Period, then all other rules are processed as if there were a Ramp Up Period of zero length.

Recovery Period:

Period at the end of the Active Interval expressed as a Duration during which the effect of the response may be reversed while the system returns to its base state. For example, a system that reduces energy use during an Event by raising the air temperature may use additional energy during the recovery period while cooling the air to the normal setting. If negative, then the Recovery Period occurs within the bounds of the Active Interval, i.e. it ends at the same moment as does the Active Interval.



IEC

Figure 4 – Time intervals of an event

The EiTarget may be used to provide information to explicitly specify the entities that apply to events. The B profile also supports signal-specific EiTargets. An EiTarget may contain one venID and one or more targets such as groupIDs, resourceIDs, or partyIDs (many others available). These may be used, for example, when a VEN is acting as an aggregator and has multiple resources behind it. Exactly how these IDs are handled is beyond the scope of this document and shall be dealt with by provisioning at the VEN and VTN. If the EiTarget element is empty, the VEN shall assume that it is the primary and only resource targeted by the events.

b) oadrCreatedEvent description

When one or more received events require a response, the VEN creates and populates an oadrCreatedEvent element and posts it to the VTN. The eiResponse element contains an application level responseCode and responseDescription and a requestID. An eiResponses element contains one or more eventResponse elements corresponding to each event. These are matched to specific events using the qualifiedEventID, which contains an eventID and modificationNumber. The optType may have a value of "optIn" or "optOut" to indicate the VENs disposition for a given event.

An initial oadrCreatedEvent response shall be sent for each event requiring a response. Subsequent oadrCreatedEvent messages may also be sent to change the opt-state of a VEN when this is allowed for a given marketContext.

The grouping of events in an oadrCreatedEvent is completely up to the VEN and does not necessarily correspond to the grouping of events in an oadrDistributeEvent. The VEN is free to send its opt-statuses by transmitting one oadrCreatedEvent payload per event or group multiple event responses into a single oadrCreatedEvent payload.

6.1.2 oadrEvent mechanism

6.1.2.1 General

- Event signal type can be SIMPLE or specific. The full set of signals described in 6.1.2.3 is supported.
- Events may contain multiple event signals.
- Event signals may contain EiTarget elements, distinct from the event-level EiTarget. However, the target types shall be constrained to endDeviceAsset with the enumerated values for device type shown in the conformance rules.

- Event signals may contain baseline elements. The rules for baseline overall duration and intervals are the same as those between the event active period and event signal intervals.
- The event description may contain `modificationDateTime` and `modificationReason` elements.
- The `EiOpt` service can be used to send schedules from the VEN to the VTN, in addition to `oadrCreatedEvent`.
- The `oadrPoll` mechanism is used in a simple HTTP PULL exchange to model the behaviour of a PUSH implementation. For more information, refer to 6.5.

6.1.2.2 Event targets and resources

This document includes several target types and also allows device class targeting to be applied at the signal level. A VEN is a communication endpoint that represents one or more logical resources (individual sheddable loads, endpoint equipment, meters, etc.). Event targets select which specific VEN resources the event applies to. If an event target is not specified, the VEN should assume that it applies to all of its resources. How resources are assigned properties (location, pnode associations, resourceIDs, groupIDs, etc.) is outside the scope of this document and is up to deployment configurations in the VTN and VEN. However, if a VEN receives an event target that it is not configured for, it should reject the message with the appropriate error code described in 6.6.

6.1.2.3 Event signal definitions

IEC 62746-10-1 allows for a diverse set of signals in the event messages. The `eiEventSignal:signalName`, `eiEventSignal:signalType`, and `eiEventSignal:itemBase` attributes are used to describe the signal. Table 2 lists standard pre-defined signals that may be used. The purpose of the pre-defined signals is to establish a common set of signals and their attributes for the purposes of interoperability. For compliance purposes, it is not a requirement that a VTN or a VEN support all the signals listed in Table 2.

Furthermore, specific deployments are free to define their own custom signals beyond what is defined in Table 2, but there are no requirements for compliance purposes that any VTNs or VENs support such signals.

The value of the `eiEventSignal:signalID` attribute is deployment-specific and primarily used to differentiate signals in cases where there can be multiple signals of the same type.

Table 2 – Signals

Signal category	Name (signalName)	Type (signalType)	Units (itemBase)	Allowed values	Description
Simple levels	SIMPLE	level	None	0,1,2,3	Simple levels
Price of electricity	ELECTRICITY_PRICE	price	currency/kWh	any	This is the cost of electricity expressed in absolute terms
	ELECTRICITY_PRICE	priceRelative	currency/kWh	any	This is a delta change to the existing price of electricity
	ELECTRICITY_PRICE	priceMultiplier	None	any	This is a multiplier to the existing cost of electricity
Price of energy	ENERGY_PRICE	price	currency/kWh	any	This is the cost of energy expressed in absolute terms
	ENERGY_PRICE	priceRelative	currency/kWh	any	This is a delta change to the existing price of energy
	ENERGY_PRICE	priceMultiplier	None	any	This is a multiplier to the existing cost of energy
Demand charge	DEMAND_CHARGE	price	currency/kW	any	This is the demand charge expressed in absolute terms
	DEMAND_CHARGE	priceRelative	currency/kW	any	This is a delta change to the existing demand charge
	DEMAND_CHARGE	priceMultiplier	None	any	This is a multiplier to the existing demand charge
Customer bid levels	BID_PRICE	price	currency/XX ^{b)}	any	This is the price that was bid by the resource
	BID_LOAD	setpoint	powerXXX ^{a)}	any	This is the amount of load reduction that was offered by a resource into a program. Positive BID_LOAD refers to curtailment, negative BID_LOAD refers to increase
	BID_ENERGY	setpoint	energyXXX ^{a)}	any	This is the amount of energy reduction that was offered by a resource into a program. Positive BID_ENERGY refers to curtailment, negative BID_ENERGY refers to increase

Signal category	Name (signalName)	Type (signalType)	Units (itemBase)	Allowed values	Description
Used to dispatch storage resources	CHARGE_STATE	setpoint	energyXXX ^{a)}	any	This is used to either charge or discharge a certain amount of energy from a storage resource until its charge state reaches a certain level
	CHARGE_STATE	delta	energyXXX ^{a)}	any	This is the delta amount of energy that should be contained in a storage resource from where it currently is
	CHARGE_STATE	multiplier	None	0,0 < 1,0	This is the percentage of full charge that the storage resource should be at
These instructions are used to set the load to values that can be expressed in terms of the desired load	LOAD_DISPATCH	setpoint	powerXXX ^{a)}	any	This is used to dispatch loads to a specific amount
	LOAD_DISPATCH	delta	powerXXX ^{a)}	any	This is used to dispatch loads to some offset from an agreed upon baseline. Note that the baseline may be the current power consumption
	LOAD_DISPATCH	multiplier	None	any	This is used to dispatch loads as some multiple of power against some agreed upon baseline. Note that the baseline may be the current power consumption
	LOAD_DISPATCH	level	powerXXX	integer value from -10 to +10	This is used to specify the load in terms of discrete levels
These instructions are used to set the load control to values that are relative to the load controller and its output capacity. This does not require the VTN or the VEN knowing precisely what the load consumption level is, but is expressed in ways that the VTN can know that the signal values will either increase or decrease the load consumption	LOAD_CONTROL	x-loadControlCapacity	None	0 % to 100 % (0,0 to 1,0)	This is an instruction for the load controller to operate at a level that is some percentage of its maximum load consumption capacity. This can be mapped to specific load controllers to do things like duty cycling. Note that 1,0 refers to 100 % consumption. In the case of simple ON/OFF type devices, then 0 = OFF and 1 = ON

Signal category	Name (signalName)	Type (signalType)	Units (itemBase)	Allowed values	Description
regardless of the specific type of device that is performing the load control. These instructions can be used for some aspects of direct load control by mapping these general instructions to specific load control commands in the VEN without the VTN needing to know precisely what device can be consuming the signal.	LOAD_CONTROL	x-loadControlLevel Offset	None	integer value, positive or negative	Discrete integer levels that are relative to normal operations where 0 is normal operations. There is no requirement to link the setpoints to specific load consumption values, but the intention is that the higher the set point the less load is consumed. Note that these are controller set points that can be mapped at the VEN side to something as simple as thermostat temperature set point offsets
	LOAD_CONTROL	x-loadControlSetpoint	None	any value	Load controller set points. There is no requirement to link the set points to specific load consumption values. Note that these are generic controller set points and can be mapped at the VEN side to something as simple as specific thermostat temperature set points.
	LOAD_CONTROL	x-loadControlPercent Offset	None	any percentage, both positive and negative	Percentage change from normal operations. The percentage does not refer to specific load consumptions values, but to load control operation levels. The lower the percentage, the less load is consumed.
a) The XXX in the table represents Real, Apparent, and Reactive versions of power or energy. b) The XX in the table represents the currency unit, such as USD.					

6.2 Report service

6.2.1 General

6.2.1.1 Overview

Subclause 6.2 describes the reporting service for IEC 62746-10-1. For detailed report content, see Annex A.

6.2.1.2 Report types

This document supports several well-known report types. Each report type is intended to represent a certain set of reporting functionality that is supported by either a VEN or a VTN that claims to support that type. Figure 5 gives the taxonomy of the different report types.

For compliance purposes, VTNs and VENs are not required to support all the different types of reports (see conformance rule 510 in Clause 9). Furthermore, for specific deployments, there can be reports used that do not conform to any of the standard report profiles, but these are not relevant for compliance purposes.

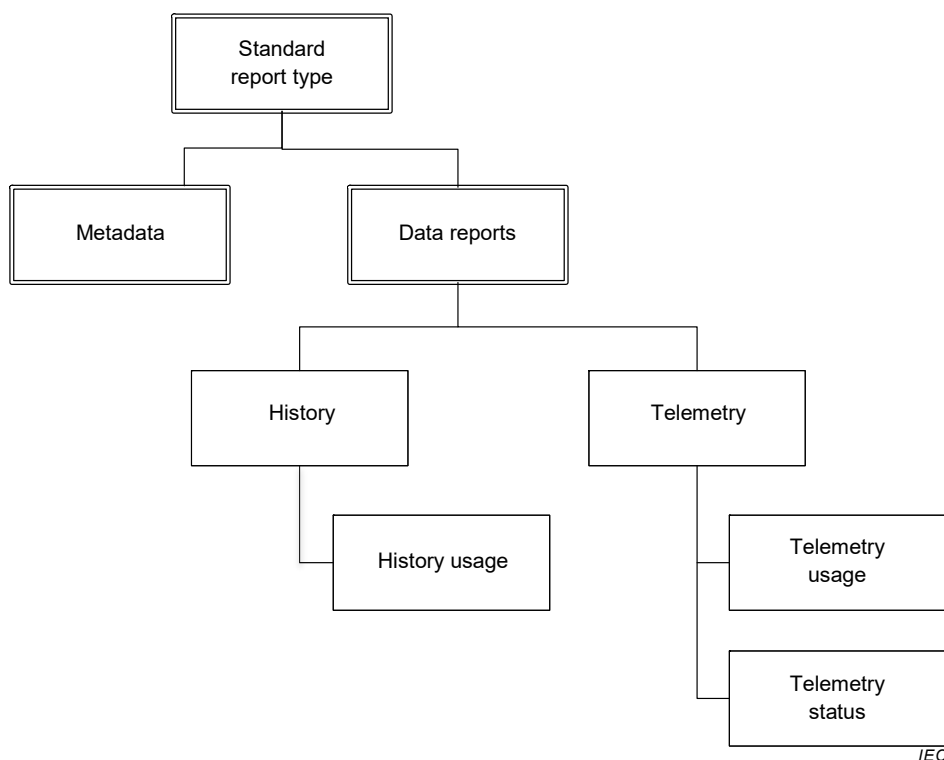


Figure 5 – Report type

Each of the report types is described in more detail below.

METADATA – This report type is used to specify reporting capabilities. It is exchanged as part of the report registration process that is described in 6.2.2.2. The METADATA report can contain a specification for one or more type of reports, each report having its own set of report descriptors and specifications. Each report in the METADATA report has a reportSpecifierID that is used in subsequent interactions to refer to that report specification. Each report specification in the METADATA report will use the reportName attribute to indicate which of the well-known report profiles it is referring to. All VTN and VEN implementations that conform to this document shall support the METADATA report and, in the case where a VTN or VEN does not support any reporting capabilities, for compliance purposes it shall still support sending the METADATA report message in order to portray the fact that it does not support any reporting capabilities.

DATA REPORTS (non-METADATA reports) – These reports are used to report actual data that can be measured or calculated. The core element of a DATA REPORT is the so-called "data point". A data point represents a certain quantity that can be measured or calculated as part of a report. Each data point has attributes such as units, etc. A DATA REPORT may contain one or more data points. The METADATA report will contain a report with a set of data points that can appear in that report and when an actual report is requested the set of data points that should appear in the report are specified by the requesting party. Each data point is represented in the schema with the rID attribute. For example, assume a VEN can measure both energy and power as two separate data points. The VEN has the choice of either specifying that it can generate two separate reports, each with a single data point or it can specify that it can generate a single DATA REPORT with two data points. The VTN would then make the appropriate report request to get the data.

There are several sub-categories of DATA REPORTS as described below.

- **HISTORY DATA REPORTS** – This is a type of DATA REPORT in which the history of the data point values is logged and can be subsequently requested. This includes the following specific types:
 - **HISTORY_USAGE** – These are logs of usage data that are typically logged by VENs and can be queried by the VTN.
- **TELEMETRY DATA REPORTS** – The term telemetry in the context of this standard refers to data that is reported periodically in real time and includes the following specific report types:
 - **TELEMETRY_USAGE** – This is usage data that is periodically reported from the VEN to the VTN in real time.
 - **TELEMETRY_STATUS** – This is the status of a resource, which may be periodically reported from the VEN to the VTN.

6.2.2 Core reporting operations

6.2.2.1 General

The reporting service supports the exchange of reports between the VEN and VTN, and vice versa. All report interactions between the VEN and the VTN are built upon the following core operations:

- registering reporting capabilities;
- requesting reports;
- sending reports;
- cancelling reports.

Figures 6 to 9 show the logical payload exchanges that support each of these operations.

6.2.2.2 Register reporting capabilities

This general use case describes how one party's reporting capabilities are registered with the other party. Report registration is performed after completion of party registration, both from the VTN to the VEN and from the VEN to the VTN. In addition, any party may send report registrations at any time after the initial registration.



Figure 6 – Interaction diagram: Register reporting capabilities

In this use case, depicted in Figure 6, the source party sends its reporting capabilities to the target party. The source party's reporting capabilities are specified using a special well-known report profile called the METADATA report, which is exchanged using the same schema as any other report.

The METADATA report contains a collection of all the different types of reports that can be sent by the source party (VEN or VTN) that is generating the report. Each reporting capability specified in the METADATA report is uniquely identified by using the reportSpecifierID attribute, which is generated by the source party when registering the report. The reportSpecifierID will allow the target party to make subsequent requests for that specific report.

Each report specified in the METADATA report is a specification for the elements and attributes that can appear in that report and is based upon one of the well-known report profiles. By specifying the metadata for the report as part of this registration process, the actual reports that can be exchanged in the future need only refer to the reportSpecifierID along with the actual data without the need to send all the other report description attributes such as units, etc.

The interaction proceeds with the following steps:

- 1) The source party first sends its reporting capabilities to the target party by using the oadrRegisterReport payload. In general, the oadrRegisterReport payload is the same as the oadrUpdateReport payload except that it contains a METADATA report. The source party sends the special well-known report of type METADATA using the oadrReport schema as described above.
- 2) The target party responds with the oadrRegisteredReport payload. The target party's response may contain an oadrReportRequest object requesting which reports the source party should generate. If there are reports that the target party knows that it wants to receive from the source party, then it should make those requests as part of this step. This is similar to requesting a report using oadrCreateReport as specified in 6.2.2.3.
- 3) If the target party requests that the source party create any reports as part of step 2), then the source party responds with the oadrCreatedReport payload.

In essence, steps 2) and 3) are equivalent to, and use the same data structures as, the interaction in which the target party requests reports from the source party using oadrCreateReport, as specified in 6.2.2.3.

Every exchange of the METADATA report shall supply all the reporting capabilities of the source party (VTN or VEN) and will therefore supplant any previously exchanged METADATA report. Furthermore, the sending of the METADATA report implicitly cancels the sending of all reports that the source party has previously requested from the target party.

Key elements included in the metadata report payloads include:

- reportSpecifierID: a unique identifier for the reporting capability, used in subsequent requests;
- rid: a unique identifier for each data point offered by the report;
- Duration: the amount of time that data can be collected (or has been collected thus far for history);
- SamplingRate.oadrMinPeriod = maximum sampling frequency;
- SamplingRate.oadrMaxPeriod = minimum sampling frequency;
- SamplingRate.onChange = whether or not data is sampled as it changes.

If the target party does not request any reports as part of step 2), then the source party should assume that there are no reports to be generated and sent to the target party. Therefore, if there are any reports that the target party has previously requested from the source party, then it shall make those requests again as part of step 2).

The source party can therefore use this interaction at any time to request that the target party notify the source party of any reports that it should create and send. This provides a mechanism for the source and target party to synchronize both the reporting capabilities of the source party and the report requests of the target party.

6.2.2.3 Request reports

This use case, depicted in Figure 7, shows how one party can request reports from the other party. Note that any reports that are being requested shall correspond to one of the reports that were previously specified in the METADATA report that was previously sent by the other party.



Figure 7 – Interaction diagram: Request reports

The source party requests a report from the target party by using the oadrCreateReport payload. That payload contains a set of reportSpecifierIDs that correspond to report capabilities in the METADATA report that was previously sent by the target party as part of the previously described oadrRegisterReport interaction (refer to 6.2.2.2). As part of the

report request, the source party specifies a set of reportRequestIDs that are used in subsequent operations on this report request.

In short, the reportSpecifierID is generated by the target during report registration (as specified in 6.2.2.2) to uniquely identify the report contents of a METADATA report, while the reportRequestID is generated by the source party of an oadrCreateReport (or oadrReportRequest object in an oadrRegisteredReport during registration) and associates the reportSpecifierID with a specific report request. For example, the source requesting a report can select only a certain subset of all data points (i.e. rIDs) that have been registered in the METADATA report identified by reportSpecifierID.

All subsequent reports sent from the target party that are in response to this request use the reportRequestID to identify the report.

The response to the oadrCreateReport payload is the oadrCreatedReport payload.

Note that the source party can only send the oadrCreateReport after the target party has sent its METADATA report as part of the reporting registration process. The exception to this is the METADATA report, which can be requested at any time by using the well-known string "METADATA" as the reportSpecifierID.

While the reportSpecifierID dictates the type of data that may appear in a report, there are some reporting parameters that are specified by the source party when making this request, including the following:

- The specific data items to appear in the report as specified by the rIDs from the corresponding METADATA report. The rID attribute is an identifier in the METADATA report that is associated with a specific type of value that may appear in the report. This is sometimes referred to as a data point and allows multiple data points to exist in a single report. The rIDs allow the source party to request a subset of possible data points that should be reported by the target party.
- Whether this is a one-time report or a recurring data stream.
- If it is a recurring data stream, the following temporal parameters are also specified: the time frame over which the report should be generated (i.e. next month, indefinitely, etc.), the time intervals at which the data points will be reported, and the frequency at which the report should be generated.
- If it is not a telemetry report, then the request specifies the span of time that the report should cover.

The meaning of dtstart in the report request defines the start time of the reporting period, while the dtstart in the returned report indicates the start time of the data collection period.

For periodic telemetry reports containing point data, if granularity and duration are the same, the report is sent immediately after the request. If granularity is smaller than duration, the delivery of the report will be delayed until enough data is collected.

Key elements included in the report request payloads include:

- Start time/Duration: when reporting should commence using granularity as sampling frequency. If Duration equals 0, it should report indefinitely.
- Granularity: Frequency of reported data. Value shall be between SamplingRate.oadrMinPeriod and SamplingRate.oadrMaxPeriod. If 0, the requester is requesting the report to use onChange (if available).
- reportBackDuration: the interval for sending reports (oadrUpdateReport or orderRegisterReport) back to the requester. If reportBackDuration == 0, then it's only a one-shot report.

See the example report request payloads in Annex E for more detailed information on the various attributes.

6.2.2.4 Send reports

This use case defines how the actual reports can be exchanged.



Figure 8 – Interaction diagram: Send reports

Figure 8 depicts the source party sending a report to the target party.

This operation can be performed by the source party only after a previous report request interaction is performed by the target party.

This operation uses the same `oadrReport` object as the report registration operation did, but is used to exchange a report with actual data elements as opposed to the METADATA report used in the registration process.

The reports sent in the `oadrUpdateReport` payload use the `EiReport` schema with the `reportRequestID` as defined by the target party in the previous report request interaction that prompted the sending of this report.

The response sent by the target party uses the `oadrUpdatedReport` payload to acknowledge receipt of the report. As part of the `oadrUpdatedReport` response, the target party may cancel the sending of any future reports using the optional `oadrCancelReport` object, which contains a list of `reportRequestIDs`. There is no confirmation of the cancellation if included in the `oadrUpdatedReport` payload, but if the report continues to be sent, the receiving party should use `oadrCancelReport` to cancel the report. In the case of periodic reports, if `reportToFollow` is set to true in the `oadrCancelReport` object by the target party, the source party is expected to send one final additional report.

See the various sample report payloads for more details on the report attributes.

6.2.2.5 Cancel reports

This interaction, depicted in Figure 9, is used by the source party to cancel ongoing (i.e. periodic) reports that are being generated by the target party.

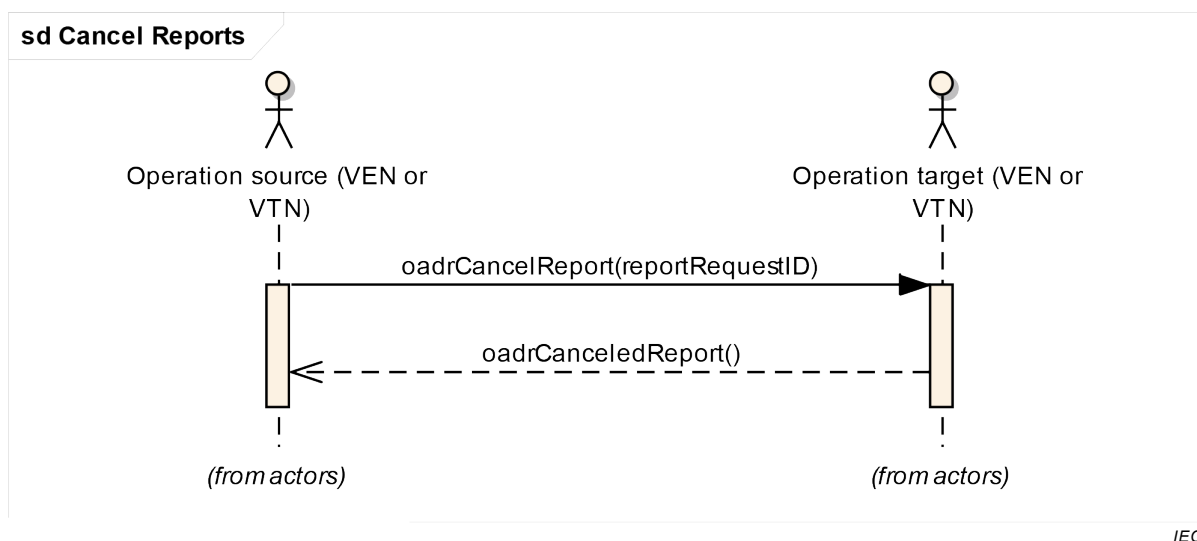


Figure 9 – Interaction diagram: Cancel reports

The source party uses the `oadrCancelReport` payload with the appropriate `reportRequestIDs` that were specified by the source party in a previous request report interaction (see 6.2.2.3). Upon receiving the `oadrCancelReport` payload, the target party stops generating and sending the reports corresponding to the `reportRequestIDs`.

The response to the `oadrCancelReport` payload is the `oadrCanceledReport` payload that is sent by the target party to acknowledge the cancellation of the report.

Note that both `oadrCanceledReport` and `oadrCreatedReport` return a list of pending reports in the `oadrPendingReports` object, which includes all reports that are scheduled for future delivery.

When cancellation of the periodic report is requested by the target party, if `reportToFollow` is set to true in `oadrCancelReport`, the source party is expected to send one final additional report.

6.3 Registration service

6.3.1 Service operations

This document uses the `EiRegisterParty` service to support in-band registration of VENs with VTNs. The following service operations (listed in Table 3) are supported:

Table 3 – EiRegisterParty payloads

Request payload	Response payload	Requestor	Responder
<code>oadrQueryRegistration</code>	<code>oadrCreatedPartyRegistration</code>	VEN	VTN
<code>oadrCreatePartyRegistration</code>	<code>oadrCreatedPartyRegistration</code>	VEN	VTN
<code>oadrCancelPartyRegistration</code>	<code>oadrCanceledPartyRegistration</code>	VEN VTN	VTN VEN
<code>oadrRequestReregistration</code>	<code>oadrResponse</code>	VTN	VEN

Registration can optionally begin with the VEN querying the VTN to determine what profiles, transports, and extensions it may support using the `oadrQueryRegistration` payload (see Figure 10). This query operation can be initiated using any of the supported transports, however the VEN will need to be configured out-of-band with the address of the VTN.

to initiate the query. The response to the query is the `oadrCreatedPartyRegistration` payload and contains information on all the profiles and transports supported by the VTN in addition to any supported extensions to the profile. The information received by the VEN can be used to determine the best configuration to use when formally registering as described in the balance of Clause 6.

Annex B provides normative information on schema extensions.

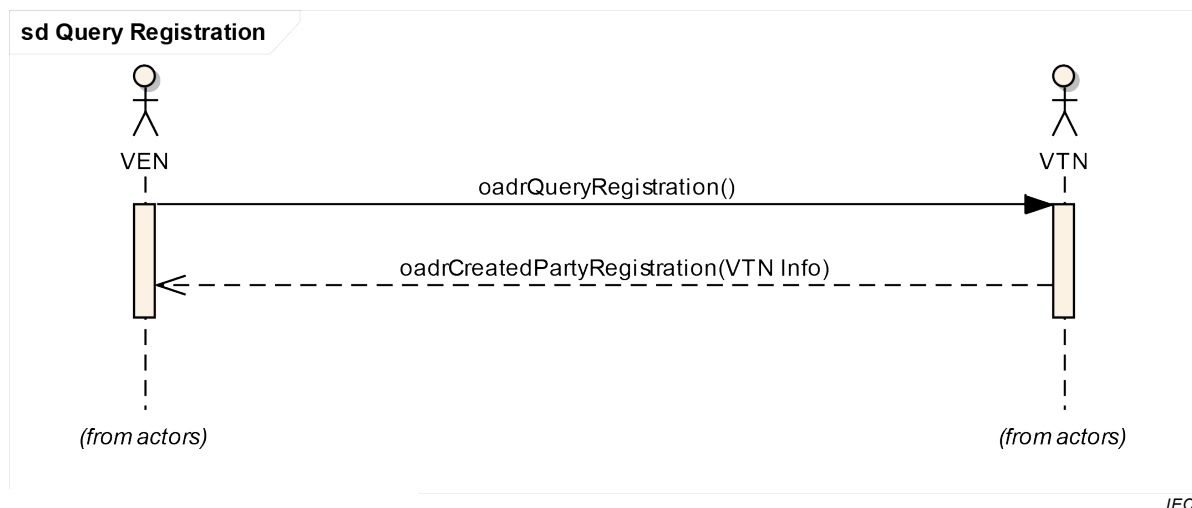


Figure 10 – Interaction diagram: Query registration

Registration is always initiated by the VEN with the `oadrCreatePartyRegistration` payload (see Figure 11). This payload provides the information on the profile and transport the VEN has decided to use for communication with the VTN, in addition to other registration-related information (see Table 4). The VTN responds with an `oadrCreatedPartyRegistration` containing all the profiles and transports supported by the VTN, IDs, and other registration-related information (see Table 5). The VTN returns a `registrationID` in its response payload, which is used for subsequent operations pertaining to this registration instance.

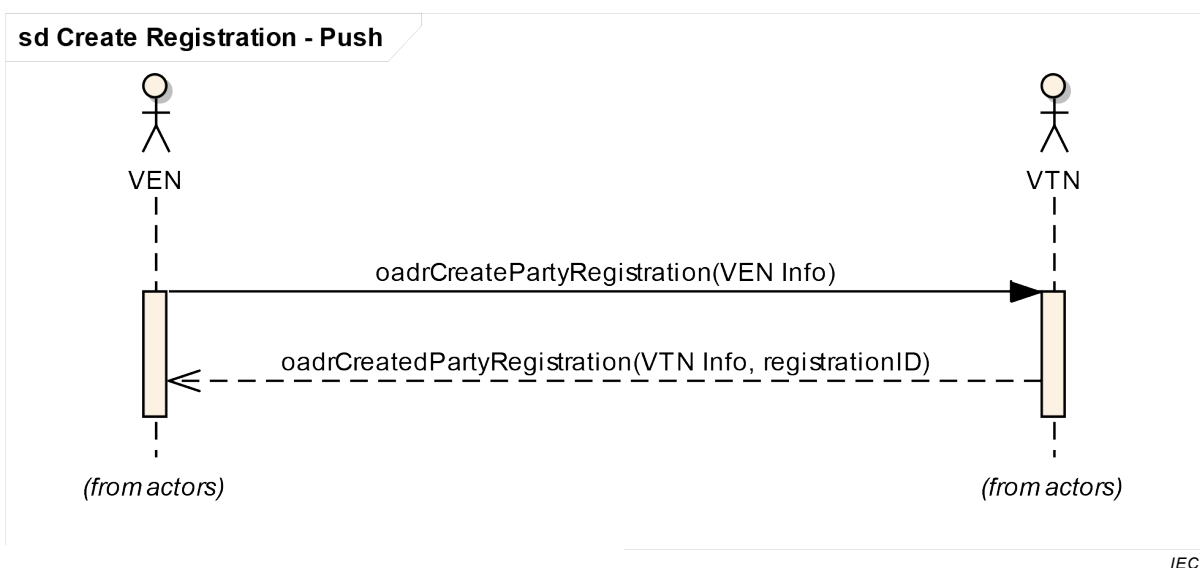


Figure 11 – Interaction diagram: Create registration

If the VEN's registration information changes, the VEN can reregister at any time using the `oadrCreatePartyRegistration` payload referencing the current `registrationID` (see Clause 2).

The same registrationID will be maintained across reregistrations until one of the parties cancels the registration.

If the VTN's registration information changes, the VTN can request the VEN to reregister using the `oadrRequestReregistration` payload. The response to this request is an `oadrResponse` acknowledgement followed by an asynchronous `oadrCreatePartyRegistration` request from the VEN.

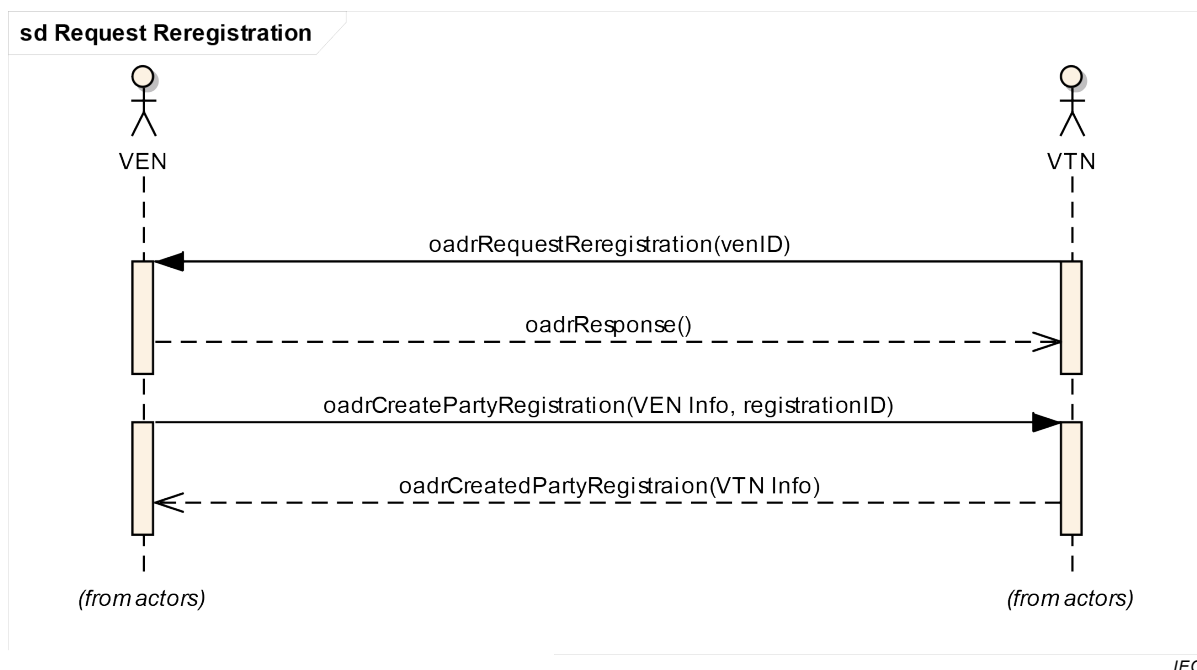


Figure 12 – Interaction diagram: Request reregistration

The VTN or VEN can cancel an active registration using the `oadrCancelPartyRegistration` payload, referencing the registrationID (see Figure 13). The other party responds with an `oadrCanceledPartyRegistration` payload.

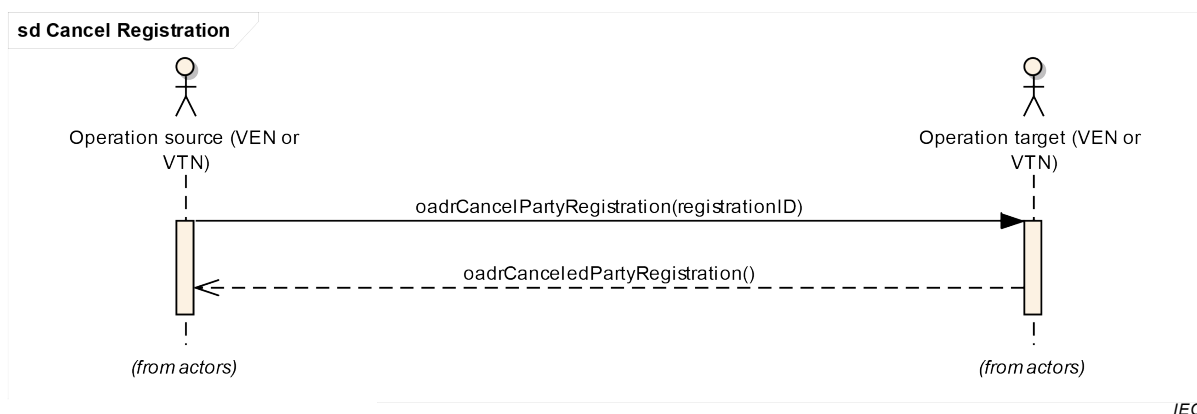


Figure 13 – Interaction diagram: Cancel registration

The sequence diagrams shown are for a PUSH interaction pattern. In an HTTP PULL model, the VEN shall periodically poll the VTN using `oadrPoll` to provide the VTN an opportunity to cancel the registration or request the VEN to reregister. Please refer to the documentation for `oadrPoll` in 6.5 for sample sequence diagrams of the PULL interaction pattern.

6.3.2 Registration information

Table 4 and Table 5 outline registration elements defined and whether the information appears in the VEN `oadrCreatePartyRegistration` request or the VTN `oadrCreatedPartyRegistration` response.

Table 4 – VEN information in `oadrCreatePartyRegistration` payload

Registration item	Comments
<code>requestID</code>	
<code>registrationID</code>	RegistrationID is only included in this payload when reregistering.
<code>venID</code>	Unless preconfigured, the <code>venID</code> is not included in the initial registration payload, but shall be present and correct for reregistration.
<code>oadrProfileName</code>	The profile the VEN selects to use for communication with the VTN. Either 2.0a or 2.0b. This document refers to 2.0b only.
<code>oadrTransportName</code>	The transport the VEN selects to use for communication with the VTN. Either XMPP or simpleHTTP.
<code>oadrTransportAddress</code>	If the selected transport is simpleHTTP or XMPP and the <code>oadrHttpPullModel</code> is false, this shall contain the address of the VEN server.
<code>oadrReportOnly</code>	This Boolean element is used to indicate the implementation is a Report Only instance of the B profile.
<code>oadrXmlSignature</code>	This Boolean element is used to indicate the implementation supports XmlSignatures.
<code>oadrVenName</code>	This is an optional human readable name for the VEN.
<code>oadrHttpPullModel</code>	This Boolean element is used to indicate the implementation uses the simpleHTTP PULL exchange model.

Table 5 – VTN information oadrCreatedPartyRegistration payload

Registration item	Comments
EiResponse Object	Contains standard payload response status information.
RegistrationID	A value used to identify the registration instance. Used for reregistration and cancelation.
venID	The venID is assigned by the VTN and returned in the registration response.
vtnID	Provided to the VEN for inclusion in payloads.
oadrProfiles:oadrProfile:oadrProfileName	A collection of one or more profiles and the associated profile name.
Nested within each profile... oadrTransports:oadrTransport:oadrTransportName	A collection of one or more transports and the associated transport name.
oadrRequestedOadrPollFreq:duration	The polling interval (i.e. the duration between two polls) using oadrPoll should be as requested by the VTN during registration in oadrRequestedOadrPollFreq. Conformance rule 500 defines the constraints for oadrRequestedOadrPollFreq.
oadrServiceSpecificInfo:oadrService:oadrServiceName oadrServiceSpecificInfo:oadrService:oadrInfo:oadrKey oadrServiceSpecificInfo:oadrService:oadrInfo:oadrValue	An object that can contain key-value pair information specific to a particular named service.
oadrExtensions:oadrExtension:oadrExtensionName oadrExtensions:oadrExtension:oadrInfo:oadrKey oadrExtensions:oadrExtension:oadrInfo:oadrValue	An object that can contain key-value pair information specific to a particular named extension.

6.4 Opt service

6.4.1 Service operations

This document specifies the EiOpt service to create and communicate Opt-In and Opt-Out schedules from the VEN to the VTN. These schedules define temporary changes in the availability, and may be combined with longer-term availability schedules and the Market Context requirements to give a complete picture of the willingness of the VEN to respond to EiEvents received by the VEN. The service operations listed in Table 6 are supported.

Table 6 – EiOpt payloads

Request payload	Response payload	Requestor	Responder
oadrCreateOpt	oadrCreatedOpt	VEN	VTN
oadrCancelOpt	oadrCanceledOpt	VEN	VTN

The VEN uses the oadrCreateOpt payload (see Figure 14) to accomplish one of the following objectives:

- to communicate to the VTN a period of temporary availability for a specific set of eiTargets,
- to communicate to the VTN a period of temporary un-availability for a specific set of eiTargets,
- to optIn or optOut of a previously acknowledged event for a specific set of eiTargets.

oadrCreateOpt payload includes an optID, which can be used in subsequent operations to reference this schedule. The VTN's response to oadrCreateOpt is an oadrCreatedOpt payload that includes an optID, which can be used in subsequent operations to reference this schedule.

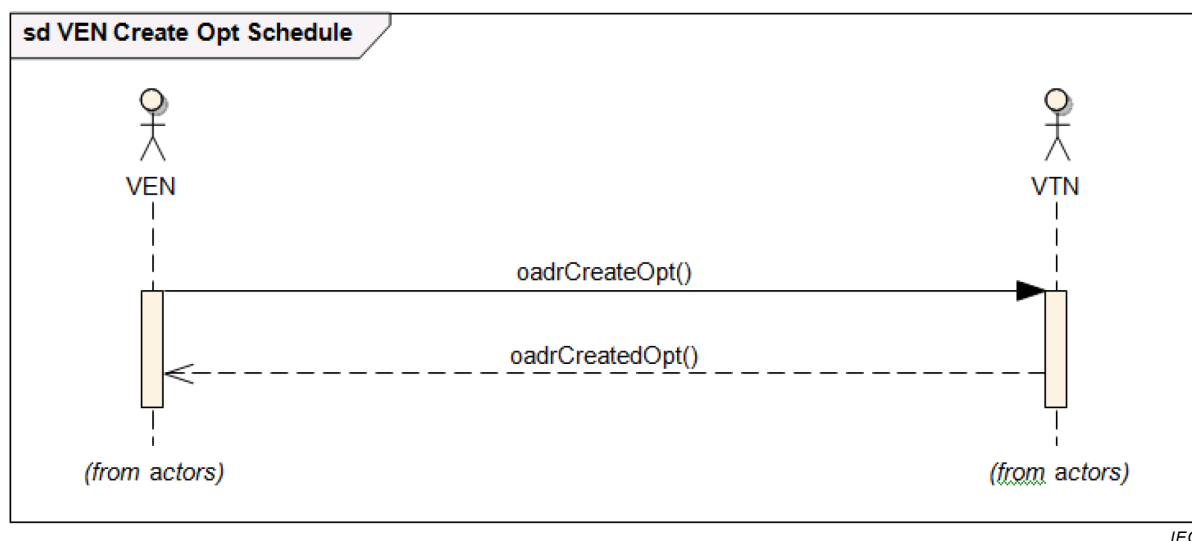


Figure 14 – Interaction diagram: Create opt

The VEN may at any time cancel a temporary availability schedule by using the oadrCancelOpt with an optID referencing the schedule to be cancelled (see Figure 15).

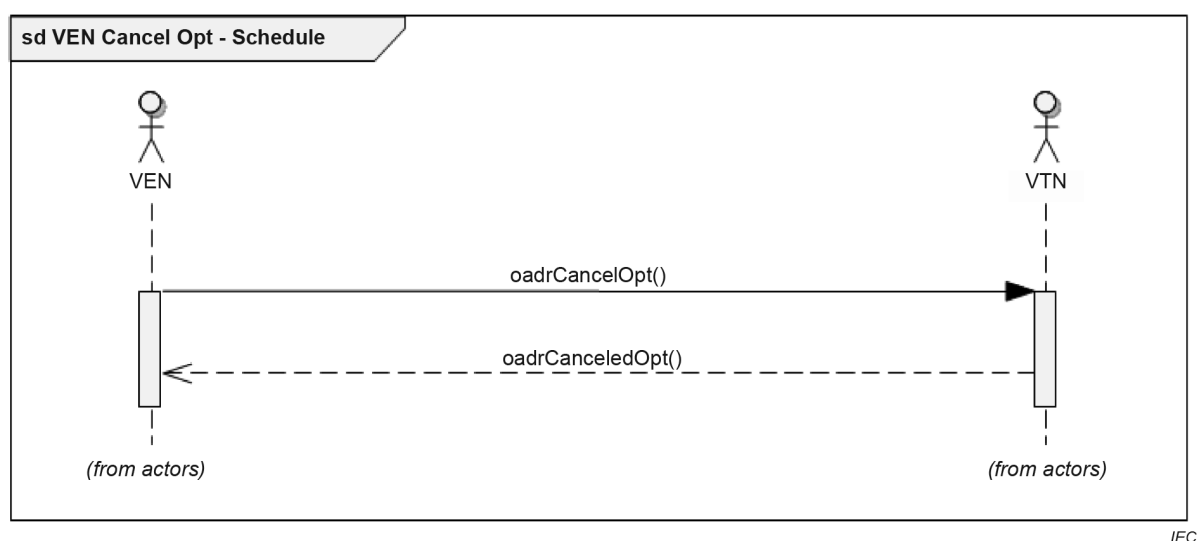


Figure 15 – Interaction diagram: Cancel opt

6.4.2 Detail requirements

If marketContext is present in the oadrCreateOpt payload, the opt schedule shall only apply to the VEN's availability with respect to events generated within the specified marketContext.

A VEN shall respond with an EiEvent oadrCreatedEvent payload in response to an oadrDistributeEvent. However, further qualifications of the optType state for the event can be accomplished by sending either an oadrCreatedEvent or an EiOpt oadrCreateOpt payload with the eventID specified. Once oadrCreateOpt is used, it will take precedence and subsequent changes to the opt state by oadrCreatedEvent will be ignored.

The oadrCreateOpt eiTarget shall contain the venID of the VEN initiating the opt schedule. If only the venID is specified in eiTarget, then the opt schedule applies to all of the resources associated with that VEN. If additional eiTarget sub-elements are defined in addition to venID, these sub-elements shall be OR'd together to define a subset of the VEN's resources.

opt schedule shall apply. Note that the VEN may send multiple opt schedules for different sets of resources identified by the eiTarget element.

A new opt schedule (optIn or optOut) sent via oadrCreateOpt shall be handled as follows when a previously sent opt schedule is still active:

- If only the venID is specified in eiTarget, the previous opt schedule with respect to future availability is replaced in its entirety by the new opt schedule for all resources associated with the VEN.
- If sub-elements are specified in eiTarget in addition to the venID, then an opt schedule shall be generated for these resources, overwriting necessarily any previously defined schedules for specific resources.
- Previously defined opt schedules for resources that are not specified by eiTarget shall remain unchanged.

Many aspects of an opt schedule are program-specific, including:

- the type of schedules that a VEN is allowed to issue: optIn, optOut, or both;
- the default opt state (optIn or optOut) for time periods not defined by either an opt or avail schedule (if defined);
- whether optIn or optOut schedules have precedence in the situation where they have overlapping time frames;
- whether overlapping time frames are allowed within the same schedule (optIn or optOut) and if they are not allowed, the expected exception handling behaviour;
- whether the VTN is allowed to send events during an optOut period.

6.5 Poll service

As new services are added, there is a need for pull-only VENs to periodically poll the VTN, particularly for VTN-initiated use cases where the information is not necessarily periodic and the VEN cannot predict when a VTN might want to send some information. In the PUSH case, this is a non-issue. However, in the PULL case, this requires the VEN to periodically poll the VTN, possibly at multiple different service endpoints, to retrieve all the information that the VTN might want to provide. oadrPoll provides a solution that allows the PULL VEN to emulate the PUSH message exchange pattern from VTN to VEN.

oadrPoll is a service-independent polling mechanism used by VENs in a PULL model to request pending service operations from the VTN. The VEN queries the poll endpoint and the VTN responds with the same message that it would have "pushed" had it been a PUSH VEN. If there are multiple messages pending a "push," the VEN will continue to query the poll endpoint until there are no new messages and the VTN responds with an eiResponse payload.

The allowed response payloads sent by the VTN in response to an oadrPoll request are:

- oadrResponse,
- oadrDistributeEvent,
- oadrCreateReport,
- oadrRegisterReport,
- oadrCancelReport,
- oadrUpdateReport,
- oadrCancelPartyRegistration,
- oadrRequestReregistration.

The rules for which payloads are valid and how those payloads are delivered are the same as if the VTN had initiated the operations and pushed the payloads to the VEN. Only one operation payload may be sent by the VTN in response to the oadrPoll message.

If a logical response is required by the VTN to the received operational payload, the VEN shall send that logical response asynchronously via a transport request. The VTN should acknowledge this logical response with an oadrResponse payload.

The VTN can optionally ignore an oadrPoll if it has not received an expected logical response to a payload delivered as a response to a previous poll. The VTN should be coded such that after some timeout it gives up waiting for the expected response and resumes responding to oadrPoll.

The oadrPoll request has a single sub-element, the venID initiating the oadrPoll. Below is an example payload:

```
<oadr:oadrPoll ei:schemaVersion="2.0b">
  <ei:venID>VEN_123</ei:venID>
</oadr:oadrPoll>
```

The following sequence diagrams in Figures 16 to 19 illustrate typical patterns using oadrPoll:

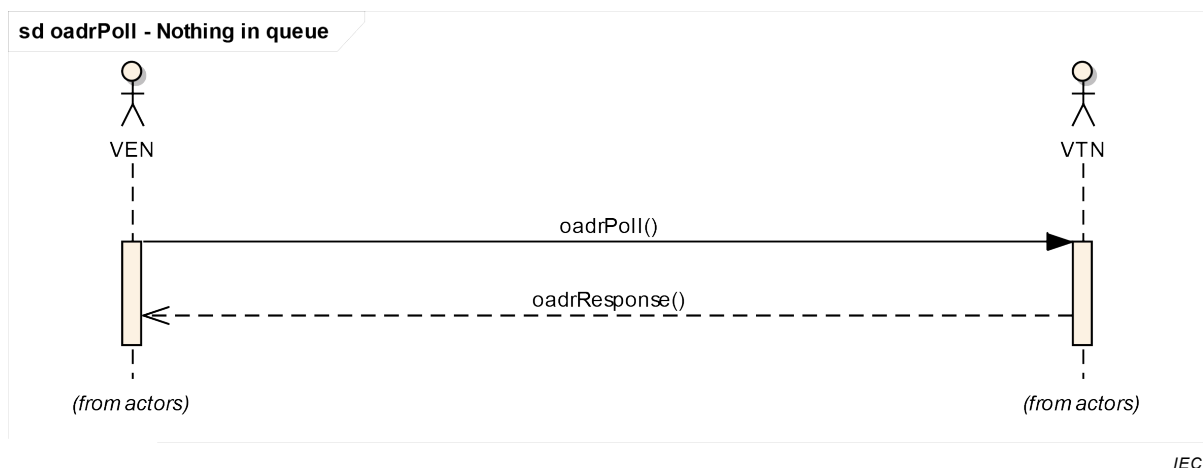


Figure 16 – Interaction diagram: oadrPoll (nothing in queue)

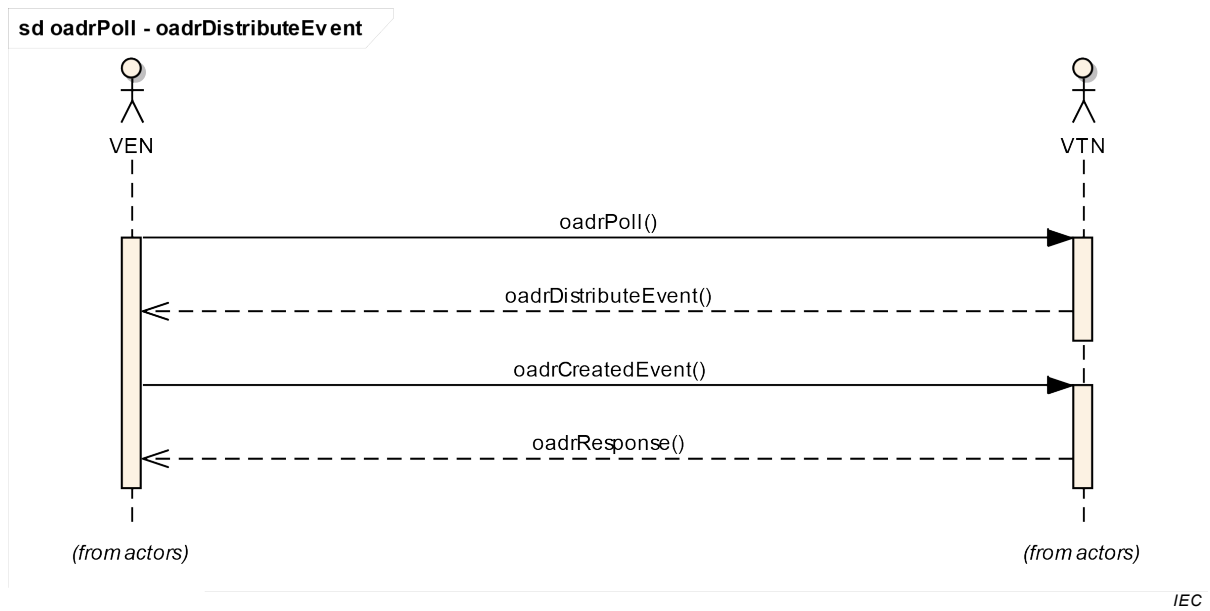


Figure 17 – Interaction diagram: oadrPoll (oadrDistributeEvent reply)

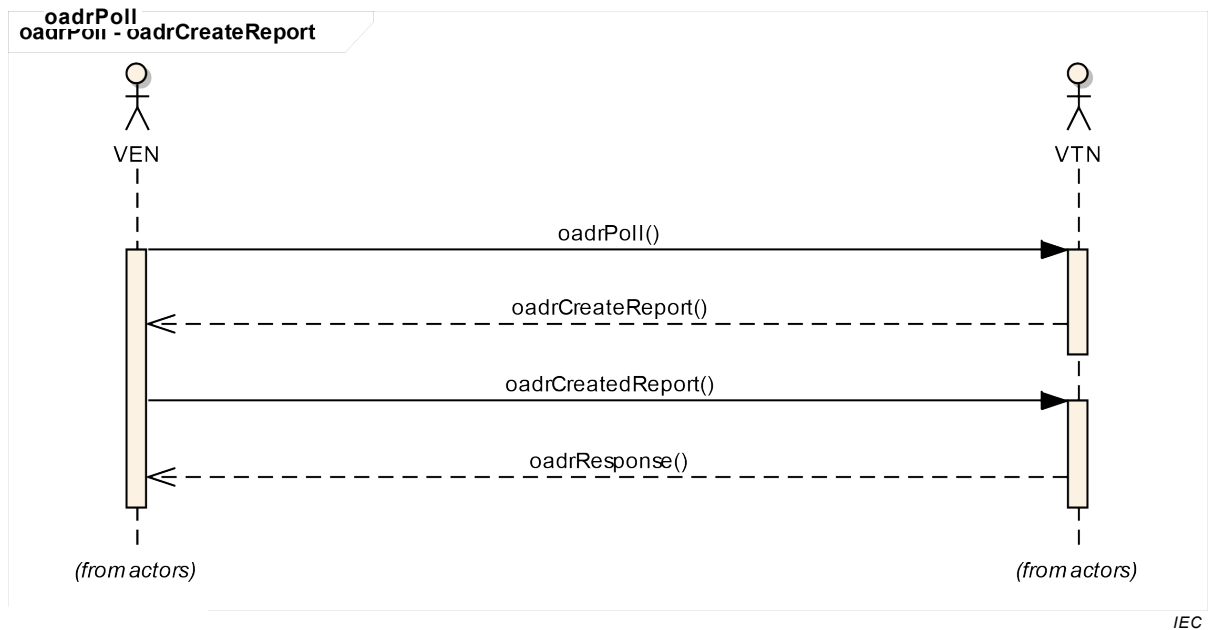


Figure 18 – Interaction diagram: oadrPoll (oadrCreateReport reply)



Figure 19 – Interaction diagram: oadrPoll (request reregistration reply)

Annex C provides several example scenarios that shall be supported by all implementations.

6.6 Application error codes

This document uses a three-digit numeric error code starting with the digit "4" to communicate application layer errors in the `responseCode` element of the various response objects in the schema. There are two general categories of errors: compliance errors and deployment errors. The second digit of the error code shall be used to differentiate between these two general categories with "5" indicating compliance errors and "6" indicating deployment errors. Error scenarios documented in the conformance rules will specify which of the error codes listed below should be used. For other error scenarios, the VEN or VTN implementer will have to use their best judgment to determine the appropriate application layer error code to use.

Compliance error codes

- 450 – Out of sequence (event ordering, uid ordering, modification number sequencing)
- 451 – Not allowed (changing an event in the past)
- 452 – Invalid ID (eventID, optID, requestID, registrationID, etc.)
- 453 – Not recognized (reportName, signalName, etc.)
- 454 – Invalid data (out of range signal or report data)
- 455 – Open
- 456 – Open
- 457 – Open
- 458 – Open
- 459 – Compliance error – Other

Deployment errors

- 460 – Signal not supported (recognized, but not supported)
- 461 – Report not supported (recognized, but not supported)
- 462 – Target mismatch (cannot resolve target/market context to VEN or its resources)
- 463 – Not registered/authorized
- 464 – Open
- 465 – Open
- 466 – Open
- 467 – Open
- 468 – Open
- 469 – Deployment error – Other

7 Transport protocol

7.1 General

This document supports a small number of transport protocols to accommodate different deployment scenarios. 2.0b VENs can either support HTTP or XMPP, or may support both. VTNs shall support both HTTP and XMPP.

7.2 Simple HTTP

7.2.1 General

Simple HTTP in this document refers to an HTTP implementation that uses HTTP POST over TLS to propagate the payloads.

7.2.2 PUSH and PULL implementation

7.2.2.1 PUSH definition

In PUSH mode, messages may be sent from the VTN to the VEN (pushed). In order to use push, the VEN shall expose HTTP URI endpoints (an HTTP server) to which the VTN may send requests such as `oadrDistributeEvent`. While this is the most efficient way to execute the communication over HTTP, it presents technical difficulties as the VEN can reside behind a network firewall.

7.2.2.2 PULL definition

In PULL mode, all operations are initiated by the VEN to the VTN. This can be thought of as a 'polling' mode, where the VEN periodically asks for updates from the VTN. The PULL mode removes the requirement for an HTTP server on the VEN, avoiding the technical limitation presented by the possibility of a network firewall in front of a VEN. However, the PULL mode has its own limitations, namely latency (due to limited polling frequency) and increased bandwidth requirements.

The PULL mode can involve a 'two-phase' execution to complete some operations. This is due to the nature of the VEN initiating the HTTP request. In a PUSH mode, the VTN can notify a VEN of a new event via the `oadrDistributeEvent` operation. The VTN would send a request with an `oadrDistributeEvent` payload, to which the VEN responds with HTTP 200 followed by an asynchronous `oadrCreatedEvent`.

However, in the PULL model, the VEN requests events from the VTN using `oadrPoll`, to which an `oadrDistributeEvent` payload is sent in the response. After parsing the response, the VEN still needs to acknowledge the creation of any new events by making a second request using the `oadrCreatedEvent` operation on the VEN.

7.2.3 Service endpoint URIs

The endpoint names will be of the form:

`https://<hostname>(:port)/(prefix/)OpenADR2/Simple/2.0b/<service>`

- "prefix" is an optional URI path prefix that can be used to separate IEC 62746-10-1 services from other services that can reside on the same HTTP server.
- "Simple" indicates the simple XML over HTTP protocol.
- <service> is the name of the service (e.g. "EiEvent", "EiReport", "EiOpt", "EiRegisterParty", "OadrPoll").

The "operation" portion of a service is defined by the XML payload sent in a request, for example, an oadrDistributeEvent payload root element specifies the oadrDistributeEvent operation.

VTN to VEN traffic via HTTP should flow through the same well-defined service endpoints used when communicating in the opposite direction. The oadrCreatePartyRegistraion:oadrTransportAddress element should contain the base address for the VEN, such as "http://myaddress:8080/prefix/". When sending traffic to the VEN, the VTN should concatenate the base address to the well-known service endpoints to form a complete endpoint address such as `http://129.6.252.49:8080/prefix/OpenADR2/Simple/2.0b/EiReport`.

Note that for implementations that expose both a VEN and VTN (such as an aggregator), these implementations shall use different URI endpoints for their VTN and VEN interfaces, for example, `https://mycompany.com/myVTN/OpenADR2/Simple/2.0b/EiEvent` and `https://mycompany.com/myVEN/OpenADR2/Simple/2.0b/EiEvent`.

7.2.4 HTTP methods

All messages shall be sent using the HTTP POST method. This helps to avoid caching and allows all operations to contain a payload in the HTTP request body.

7.2.5 Failure conditions

The following failures can occur for a given operation:

- TCP (or below) fails;
- TLS negotiation fails;
- HTTP fails (HTTP error code);
- application acknowledgement fails (application error code);
- response failure (timeout or application error code).

The proper action for each failure condition depends upon the application and the operation being attempted. Since all operations are idempotent, it is safe to retry any operation.

In the case of TCP failure, it is always recommended to retry the operation.

Application-level errors are defined in 6.6.

7.2.6 HTTP response codes

The following HTTP status codes, defined in IETF RFC 2616, are used in this document:

200 OK – any response that the endpoint was able to handle completely and send a valid response payload. This includes responses that can indicate an error at the application level.

(e.g. 'you gave me an invalid event ID'). Errors that indicate a failure at the transport level are handled by transport-level HTTP error codes.

404 Not Found – the VEN or VTN does not support the requested operation. The requestor shall not re-send the request.

406 Not Acceptable – if a payload is sent that does not validate against the schema, or if a request content-type is unsupported. The requestor shall not re-send the request without first modifying it.

501 Not Implemented – if any request is made with an unsupported HTTP method. The requestor shall not re-send the request without fixing the HTTP method.

503 Service Unavailable – indicates that the server is temporarily unavailable, possibly due to inability to handle the current request load. This error in particular shall indicate to the requestor that it shall execute quiesce logic in order to not put further strain on the server. The requestor shall retry the request after the proper quiesce period.

500 Internal Server Error – undefined or unexpected server error. The requestor can retry the request after a quiesce period.

For all error (non-200) codes, the content body of the response is undefined. The server can choose to send some informational message in the response, but the requestor is not obligated to parse or understand it.

All application-level error conditions are conveyed through the status code element of oadrResponse payload.

7.2.7 Message timeouts

There are no prescribed connect or response timeout thresholds for this document. Implementations shall allow configurable timeouts in order to handle IP networks with different latency characteristics. HTTP clients shall use a request timeout of at least 5 s.

7.2.8 Message retry/quiesce behaviour

When a request fails for any reason (either due to physical or network-level failure or a timeout), the requestor shall institute 'back-off' or quiesce logic to avoid flooding the network or receiver with requests.

Clients shall begin quiesce at some small interval (say, 1 s) plus or minus some random 'jitter,' which is a small percentage of that interval (say, 10 %). For example, the first quiesce interval for a device might be between 0,9 s and 1,1 s. Then the device may retry the request. If subsequent retries fail, the quiesce interval shall double from the prior interval, again adding a random jitter of $\pm 10\%$ of that interval. This doubling for subsequent failures shall continue up to some maximum, probably dependent on the poll interval in the case of a VEN polling a VTN. This is known as a "truncated binary exponential back off algorithm."

7.2.9 PULL timing

The ability to configure the poll interval for a PULL mode is not defined in this document, however VEN implementations shall allow the poll interval to be configurable at a millisecond resolution. This adjustment resolution is only mandatory for the communication protocol and this requirement does not infer that building systems require millisecond-level control. Responsiveness of building systems is program-dependent.

In this document, a VTN may configure the minimum poll interval for the VEN by setting oadrCreatedPartyRegistration:oadrRequestedOadrPollFreq accordingly.

VENs shall also be configurable to induce some 'jitter' – a random offset from the absolute poll interval – in order to avoid request spikes on the VTN caused by many VENs initiating a PULL request at the exact same instant within the poll interval.

7.2.10 HTTP headers

7.2.10.1 General

The HTTP headers in 7.2.10.2 to 7.2.10.8 shall appear in requests or responses (where indicated):

7.2.10.2 Accept

The accept request header specifies the expected content-type of a response. Since responses are always "application/xml", the accept header may be omitted. However, if it is included, the value of the accept request header shall always be "application/xml".

7.2.10.3 Accept-Encoding

This request header indicates if a client supports content compression of the response payload. A VEN may include this header in a request if it supports content compression such as gzip or deflate. If the VEN includes this header, the VTN shall honor it and compress the response content using one of the methods given in the request header.

7.2.10.4 Content-Encoding

If a VTN is responding to a request for which it has a compressed payload, it shall include a content-encoding response header indicating the correct encoding method, such as gzip or deflate.

PUSH operations from a VTN shall not utilize the content-encoding header in the request, since it would require the VTN to have a priori knowledge of which content-encodings are supported by each and every VEN.

7.2.10.5 Content-Length

The content-length header shall be used according to IETF RFC 2616 to indicate content body size of all request and response payloads.

Note that 'chunked' transfer encoding (where content-length is unknown) shall not be used in this document. Implementers shall assume that the total content body length is known when the response headers are sent, and shall not send chunked responses.

7.2.10.6 Content-Type

This shall be used for request and response messages, indicating payload MIME type. The appropriate value is "application/xml". The content-type may also specify a character encoding. For this document, the only supported character encoding is UTF-8. If a charset is included, the entire header value shall appear as "application/xml; charset=utf-8".

7.2.10.7 Host

The Host header shall be included in all requests in accordance with IETF RFC 2616, section 14.23 (3).

7.2.10.8 User-agent

The requestor may include the User-Agent header, but its presence shall not be relied upon, nor shall it materially affect the behaviour of the server handling the request.

7.3 Transport-specific security

7.3.1 General

TLS shall be used to encrypt all traffic. The client shall always validate the server's TLS certificate given during the handshake.

7.3.2 TLS client certificate

Client certificates shall be used for mutual HTTP client authentication. The entity initiating the request (the client) shall have an X.509 certificate that is validated by the server during the TLS handshake. If no client certificate is supplied, or if the certificate is not valid (e.g. it is not signed by a trusted CA, or it is expired), the server shall terminate the connection during the TLS handshake.

If the certificate appears valid during the TLS handshake, the connection is established and the HTTP request proceeds. Once the server receives the HTTP request, it shall perform authentication, given the credentials in the client certificate. The VTN shall use the certificate public key as the primary identification mechanism and not the identity in the certificate. The client credentials shall be compared to ensure they match the venID that appears in the payload of the client request. If credentials do not match, or if the server otherwise determines that the request is not authorized, it shall respond with an application layer 463 error in the eiResponse object of an appropriate message (e.g. in an oadrCreatedPartyRegistration response if the request from the VEN was an oadrCreatePartyRegistration).

To implement this document, any acceptable security certificate can be used as supported by the DR program operator. One option is to use security certificates that are regulated by the OpenADR Alliance Certificate Policy¹.

7.4 XMPP

7.4.1 General

XMPP is a stateful, bi-directional protocol ideal for transmitting messages in XML format. The core protocol is specified in IETF RFC 6120, and addressing is defined in IETF RFC 6122 and is also referenced in Annex G of 61968-100:2013. RFC 6120 section 13.7 describes certificate-based authentication handling. It should be noted that in the intended implementation of this document, the XMPP server is part of the VTN server infrastructure for security purposes. If the XMPP server resides outside the VTN's security framework, additional security measures may be necessary.

7.4.2 Exchange model implementation

By nature, XMPP is a bi-directional, stateful protocol. As such, any client using XMPP can implement both PUSH and PULL operations seamlessly, with the exception of oadrPoll, which is disallowed over XMPP. PULL operations are simply those initiated by the VEN, and PUSH operations are initiated by the VTN.

7.4.3 Service endpoints

A jabber identifier (JID), which is similar to an email address, defines both VTN and VEN service endpoints. The fully qualified JID performs the same function as an endpoint URI in an HTTP implementation. The definition of JIDs is further described in 7.4.5.1.

¹ OpenADR Alliance Certificate Policy is a publication of the OpenADR Alliance. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

7.4.4 Service execution

Because XMPP is a message-based protocol, execution of services occurs by passing XMPP messages that contain an XML payload.

7.4.5 Implementation of XMPP features

7.4.5.1 JIDs

While the prospect of using multiple resources per JID opens some interesting possibilities (e.g. exposing each building at a facility as a different resource), this strategy should be used with some caution, as the same mechanism is not supported by other transports such as HTTP.

VEN clients should always authenticate with a fully qualified JID (rather than allow the server to assign a random resource ID.)

VTN clients can choose to expose the operations either as an XMPP service or as a client JID. Although most service operations will be 'pushed' from VTN to VEN, a VEN can still need to send some requests to the VTN. Rather than use some configuration to define the JID for VTN services, services shall be discovered at runtime via service discovery (described in 7.4.5.9).

VTN to VEN traffic via XMPP shall flow through an endpoint defined by a single fully qualified JID. This VEN JID will be communicated to the VTN as part of the `oadrCreatePartyRegistration` payload in the `oadrTransportAddress` element. The resource name used for the JID can be arbitrary, such as "client". For instance, the contents of the `oadrTransportAddress` element might be something like "ven@domain/client".

VEN to VEN traffic is disallowed and shall be filtered by the XMPP server.

7.4.5.2 Use of the packet 'type' attribute

All operations shall use the 'set' IQ type attribute for all service operations.

7.4.5.3 Use of message versus IQ packets

Any service operations that require an application-level response (e.g. a response payload) or a transport-level response code shall use an IQ packet. The XMPP protocol then mandates a response IQ from the recipient.

Any service operations that do not require a response, such as a broadcast operation (or possibly feedback), may use a message packet to send the payload. Handling of the packet should be identical to if it were an IQ, except that the recipient sends no response. This is a bandwidth optimization for use cases where a response is not expected, and a response IQ would be unnecessary.

7.4.5.4 Error handling

7.4.5.4.1 Transport-level status codes

Under normal conditions, an IQ response will have a `type='result'` attribute. This is analogous to an HTTP 200 response. Service operations that define an empty HTTP 200 response will simply look like an IQ element with no child payload element:

```
<iq type='result'
  to='ven1234@xmpp.myvo.net/client'
  from='vtn.xmpp.somevtn.net'
  id='1' />
```

7.4.5.4.2 Transport-level errors

Operations that result in a transport-level error will have a type='error' and a child <error> element that indicates the transport-level status code as described in IETF RFC 6120, section 8.3. For example:

```
<iq type='error'
  to='ven1234@xmpp.myco.net/client'
  from='vtn.xmpp.somevtn.net'
  id='2'>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

Since XMPP does not use numeric status codes, direct mapping of HTTP status codes is not possible, and logic for handling transport-level errors is outside the scope of this document. Guidelines for handling different types of XMPP errors can be found in IETF RFC 6120, section 8.3.3.

If a client receives a payload that does not adhere to the schema, a client should respond with a bad-request error as defined in IETF RFC 6120, section 8.3.3.1.

Note that application-level errors shall be handled the same as in HTTP. That is, a normal 'result' XMPP response should be returned with the application-level error details in the payload.

7.4.5.5 Presence

VEN clients shall use presence packets to notify subscribed entities of online and offline status as described in IETF RFC 6121, section 4. All other status broadcasts from a client should be considered informational.

After a VEN completes session negotiation, it shall send an 'available' presence stanza as defined in IETF RFC 6121, section 4.2.1:

```
<presence />
```

If a VEN deliberately terminates its XMPP connection (e.g. owing to a graceful shutdown, not an unexpected, sudden connectivity loss), it shall first send an 'unavailable' presence stanza as defined in IETF RFC 6121, section 4.5.1:

```
<presence type='unavailable' />
```

The VTN should not generally advertise its presence to the VENs. A VTN could subscribe to VENs and receive presence updates, but can cause scalability issues and should be used with caution.

7.4.5.6 XMPP ping

Support for XMPP ping (as specified in XMPP Standards Foundation specification XEP-0199 version 2) is mandated from the VEN to the XMPP server. The VEN can notice disconnection from the XMPP server and, for example, re-establish connection or send an alarm if one or more pings are not replied to within a certain time. XMPP ping intervals and other parameters related to XMPP ping may be set locally, and exchange of these parameters is not required as part of registration.

7.4.5.7 Authentication

All clients shall support SSL/TLS and authentication as defined in IETF RFC 6120, sections 13.8 and 13.9.4. Clients shall also implement simple authentication and security layer SASL EXTERNAL in order to use certificate authentication as defined in IETF RFC 6120.

The username of the VTN/VEN logging in to the XMPP server shall match the Common Name (CN) field of the x509 certificate that is used in the authentication process.

7.4.5.8 Rosters

While rosters are a core feature of XMPP and commonly occur during XMPP session initiation, they have no defined role within this document and, therefore, their use is undefined within the scope of this document. A roster request may be performed by a client; however, under most circumstances, the server can simply respond with an empty roster. Optimally, a client (particularly VEN clients) should not request a roster from the server for the purposes of this document.

More specifically, VEN clients should not typically add other entities to their roster, as this will result in additional network overhead owing to presence broadcasts.

VTN clients might choose to use the roster as a mechanism to track the presence of online VEN clients, however this is not strictly a feature or requirement of this document.

7.4.5.9 Service discovery

7.4.5.9.1 General

XMPP service discovery XEP-0030 shall be used to define the address for services on the VTN. This allows the VTN flexibility in how it implements services, while removing the need for additional configuration on the VEN, which would be outside the scope of this document.

7.4.5.9.2 Discovery of the features

After a VEN has completed session initiation, it may perform an XEP-0030 'info' query to the bare domain to which the VEN has authenticated. For example, if a VEN has connected as 'ven1@xmpp.myco.net/client' then its initial service discovery query would look like the following:

```
<iq type='get'
  from='ven1@xmpp.myco.net/client'
  to='xmpp.somevttn.net'
  id='info1'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

To which the server would respond:

```
<iq type='result'
  from='xmpp.somevttn.net'
  to='ven1@xmpp.myco.net/client'
  id='info1'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <feature var='http://openadr.org/openadr2'/>
    <feature var='http://jabber.org/protocol/disco'/>
  </query>
</iq>
```

This indicates, in an XMPP-compliant fashion that the XMPP server supports the protocol.