ISO/IEC 15067-3-31

Edition 1.0 2024-03

INTERNATIONAL STANDARD

colour inside

Information technology – Home Electronic System (HES) application model – Part 3-31: Protocol of energy management agents for demand-response energy management and interactions among these agents

ISO/IEC 15067-3-31:2024-03(en)

IEC Secretariat
3, rue de Varembé
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 corrigendum or an amendment might have been published.

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 It also gives information on projects, replaced and withdrawn publications.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 once a month by email.

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.

IEC Products & Services Portal - products.iec.ch
Discover our powerful search engine and read freely all the publications previews, graphical symbols and the glossary. With a subscription you will always have access to up to date content tailored to your needs.

Electropedia - www.electropedia.org
The world's leading online dictionary on electrotechnology, containing more than 22 500 terminological entries in English and French, with equivalent terms in 25 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

ISO/IEC 15067-3-31

Edition 1.0 2024-03

INTERNATIONAL STANDARD

colour inside

Information technology – Home Electronic System (HES) application model – Part 3-31: Protocol of energy management agents for demand-response energy management and interactions among these agents

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 35.200

ISBN 978-2-8322-8357-8

CONTENTS

INFORMATION TECHNOLOGY – HOME ELECTRONIC SYSTEM (HES) APPLICATION MODEL –

Part 3-31: Protocol of energy management agents for demand-response energy management and interactions among these agents

FOREWORD

1) ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.
2) The formal decisions or agreements of IEC and ISO 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 and ISO National bodies.
3) IEC and ISO documents have the form of recommendations for international use and are accepted by IEC and ISO National bodies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC and ISO documents is accurate, IEC and ISO 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 and ISO National bodies undertake to apply IEC and ISO documents transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC and ISO document and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC and ISO do not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC and ISO marks of conformity. IEC and ISO are not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this document.
7) No liability shall attach to IEC and ISO or their directors, employees, servants or agents including individual experts and members of its technical committees and IEC and ISO National bodies 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 ISO/IEC document or any other IEC and ISO documents.
8) Attention is drawn to the Normative references cited in this document. Use of the referenced publications is indispensable for the correct application of this document.
9) IEC and ISO draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). IEC and ISO take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, IEC and ISO had not received notice of (a) patent(s), which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at https://patents.iec.ch and www.iso.org/patents. IEC and ISO shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 15067-3-31 has been prepared by subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology. It is an International Standard.

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

Draft	Report on voting
JTC1-SC25/3205/FDIS	JTC1-SC25/3219/RVD

Full information on the voting for its approval can be found in the report on voting indicated in the above table.

The language used for the development of this International Standard is English.

This document was drafted in accordance with ISO/IEC Directives, Part 2, and developed in accordance with ISO/IEC Directives, Part 1, and the ISO/IEC Directives, JTC 1 Supplement available at www.iec.ch/members_experts/refdocs and www.iso.org/directives.

A list of all parts of the ISO/IEC 15067 series, published under the general title *Information technology – Home Electronic System (HES) application model*, can be found on the IEC and ISO websites.

IMPORTANT – The "colour inside" logo on the cover page of this document 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

This document specifies a protocol to facilitate communications among interacting energy management agents (EMAs) for energy management applications. This document specifies a message format and sequences of message exchanges in a protocol called the EMA protocol (EMAP) to enable logical interactions among EMAs.

EMAP includes messages that enable demand response (DR) in community housing such as an apartment building or a campus of houses or apartment buildings, as introduced in ISO/IEC 15067-3-3. ISO/IEC 15067-3-3 was created because utilities throughout the world are investing heavily in smart grid infrastructures to ensure a reliable supply of electricity and to accommodate DR technologies for residential homes and apartment buildings. DR programmes are being offered to residential consumers for energy conservation and for energy management to align demand for power with available supplies according to customer preferences for appliance usage and budget constraints. An interacting EMA extends the capability of a single EMA to allocate energy among houses efficiently in a community and among appliances within houses, and to accommodate a choice of external energy sources or local energy sources or both linked to an EMA. External sources can be public utilities or distributed energy resources (DERs) in other homes, possibly purchased using transactive energy. Local sources can include renewable power generators and storage devices at the customer premises linked to an EMA. Consumer devices linked to an EMA can participate in energy management programmes such as DR and can interconnect logically via an EMA with local DER equipment such as generators (wind and solar) and energy storage devices.

This document facilitates utility-based DR programmes, but does not mandate such programmes. Consumers may choose to provide electricity using DER locally with a house, an apartment complex, a community, or a microgrid without any connection to a public utility. This document also applies to non-utility DR programmes that are based within the apartment complex operating as a microgrid.

Typical smart energy services can include integrated energy management for multiple energy systems, energy sharing and trading within the community, energy information sharing for more efficient energy usage, etc. These energy services offer benefits in electrical energy management in a house, a residential community or a building consisting of multiple apartments.

The intent of EMAP is to accommodate flexible and efficient energy management systems over a broad range of EMA deployments. This document has been developed to promote interoperability among products from different manufacturers. EMAP enables automated demand-response services in a house, a residential community or a building consisting of multiple apartments for co-ordinating and allocating energy consumption and generation among multiple EMAs in different locations. The co-ordination among EMAs offers improved energy management and overall efficiency. Each EMA enables the allocation of energy among appliances and switching energy sources from grid to local generation or storage according to consumer preferences.

This document specifies message formats for DR, pricing, and DER communications to manage customer energy resources, including load, generation, and storage in a home, building and apartment complex. Communication messages specified in this document for the DR command set support direct load control, time-of-use (TOU), critical-peak-pricing (CPP), real-time pricing (RTP), peak time rebates, various types of block rates, transactive energy, and a range of opt-in, opt-out and service modifications. This document can interact with IEC 62746-10-1, which specifies message and application-layer protocol profiles relevant for systems connected to an external DR service provider. Unlike IEC 62746-10-1, EMAP can support bi-directional exchange of DR events between EMAs for co-operative energy management according to the customer's budget by using the opt commands in a hierarchical or point-to-point architecture.

If a logical hierarchical tree structure has been installed, EMAP messages (e.g. report, opt) may be transferred with aggregation or disaggregation through intermediate EMAs. In addition, the messages (e.g. DR command, report) specified in this document may be relayed across intermediate EMAs. The aggregation/disaggregation or relay mode is optionally provided.

INFORMATION TECHNOLOGY – HOME ELECTRONIC SYSTEM (HES) APPLICATION MODEL –

Part 3-31: Protocol of energy management agents for demand-response energy management and interactions among these agents

1 Scope

This document specifies a protocol for energy management agents (EMAs) to facilitate communications among these agents for demand response (DR) energy management applications. The EMA protocol (EMAP) provides a logical connection among EMAs in community housing such as an apartment building or a campus of houses or apartment buildings. This document also specifies interaction procedures and message formats for DR energy management as introduced in ISO/IEC 15067-3-3. The EMAP supports interactions among EMAs at OSI (Open System Interconnection) layer 7 with a message transfer protocol. An EMA can be embedded in devices such as a thermostat, a smart appliance, or other consumer products. The choice of interconnection depends on the system and the network topology, which can be arranged in a mesh or hierarchical tree structure. An intermediate EMA may relay messages sent between EMAs.

NOTE 1 The application message set that uses the EMAP OSI layer 7 is not part of the protocol specification in this document. It is part of the EMA above OSI layer 7. The application message set can use the ISO/IEC 14543 series, ISO/IEC 10192-3, the IEC 62541 series and the ISO/IEC 30118 series.

NOTE 2 This document facilitates utility-based DR programmes, but does not mandate such programmes. Consumers can choose to provide electricity using DER locally with a house, an apartment complex, a community, or a microgrid without any connection to a public utility.

NOTE 3 In a residential microgrid, data such as DR messages and price signals can be issued by an energy management system or an EMA for scheduling, forecasting functions, etc. to optimize energy consumption or generation without relying on a public utility.

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/IEC 15067-3:2012, *Information technology – Home Electronic System (HES) application model – Part 3: Model of a demand-response energy management system for HES*

ISO/IEC 15067-3-3:2019, *Information technology – Home Electronic System (HES) application model – Part 3-3: Model of a system of interacting energy management agents (EMAs) for demand-response energy management*

IETF RFC 7252, *The Constrained Application Protocol (CoAP)*, edited by Z. Shelby et al., June 2014, available at: https://tools.ietf.org/rfc/rfc7252.txt [viewed 2023-05-31]

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 15067-3:2012, ISO/IEC 15067-3-3:2019 and the following apply.

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

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

3.1.1
client EMA
cEMA
energy management agent (EMA) that acts as a client to another EMA

[SOURCE ISO/IEC 15067-3-3:2019, 3.1.1]

3.1.2
EMA protocol
EMAP
protocol to facilitate communications among interacting energy management agents (EMAs) for energy management applications

3.1.3
transaction
smallest unit of a work process consisting of an exchange between two or more participants or systems

[SOURCE ISO 15489-1:2016, 3.18]

3.1.4
server EMA
sEMA
energy management agent (EMA) that acts as a server to other EMAs

[SOURCE ISO/IEC 15067-3-3:2019, 3.1.7]

3.2 Abbreviated terms

cEMA	client EMA
CoAP	constrained application protocol
DER	distributed energy resources
DR	demand response
DTLS	datagram transport layer security
EMA	energy management agent
EMAP	EMA protocol
HAN	home area network
HTTP	hypertext transfer protocol
JSON	javascript object notation
OSI	open system interconnection
sEMA	server EMA
UDP	user datagram protocol

4 Conformance

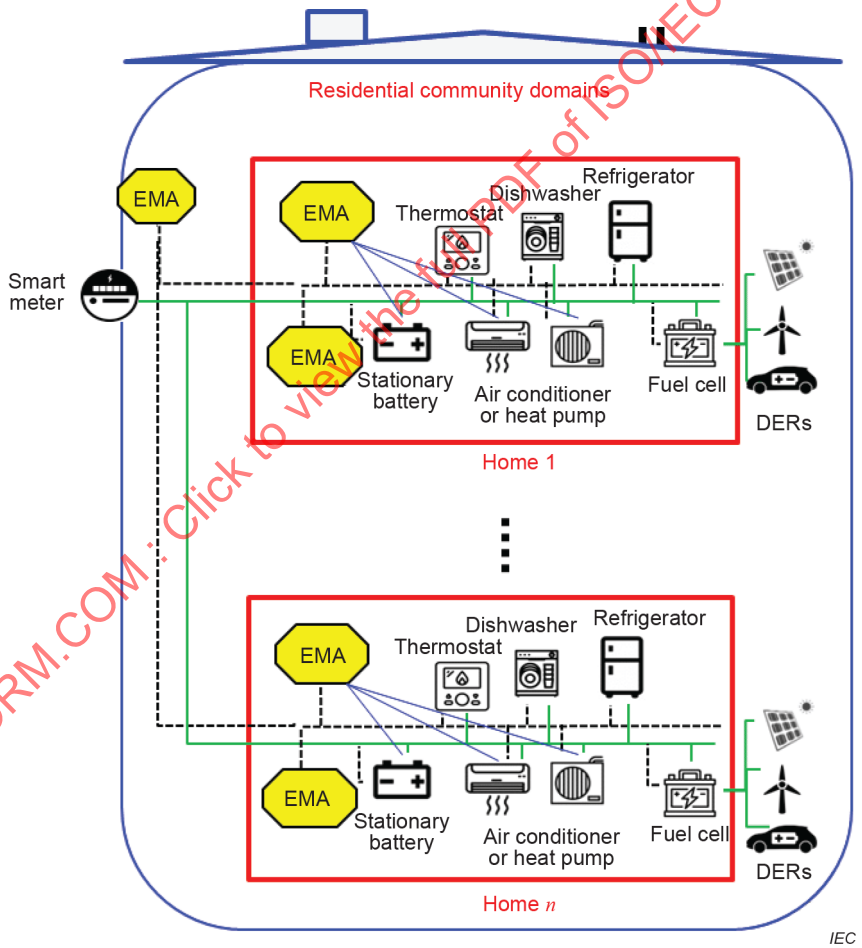
Implementations of a protocol among EMAs shall conform to protocol procedures and message formats specified in Clauses 6 and 7.

5 Energy management agent architecture for home or residential community

5.1 Overview

Figure 1 shows an example of a physical energy management system in a community that includes multiple private residential homes and common spaces. The configuration shown in Figure 1 consists of multiple interacting EMAs in the home, which are specified in ISO/IEC 15067-3-3.

In some cases, one or more EMAs may be physically located in a home. For an apartment complex, there may be an EMA in each apartment, in the common area of each building, and in the management office of the complex. An EMA may be embedded in devices such as a thermostat, a smart appliance, or other consumer products.



NOTE 1 Adapted from Figure 1 of ISO/IEC 15067-3-3:2019 to clarify the functions of the EMAs.

NOTE 2 The home area networks (HANs) are shown with dashed line in black and the power line is shown in solid green.

NOTE 3 The shapes are explained in 5.2 of ISO/IEC 15067-3-3:2019.

Figure 1 – Example of an energy management system in a residential area

The EMA applies complex algorithms to exchange energy-related data among EMAs and devices. Such energy-related data may include energy consumption targets, cost of energy, usage data, energy allocation to appliances, distributed energy resources (DERs), and EMA locations and rolls within the community.

The EMAP supports interactions among EMAs at OSI layer 7 including application messages and data. The interactions provide the capability of a system with a single EMA to allocate energy among houses efficiently in a community and among appliances within houses, and to accommodate a choice of external energy sources or local energy sources or both linked to the EMA. External sources may be public utilities or other suppliers. Local sources may include local power generators and storage devices linked to the EMA. Consumer devices managed by the EMA can also participate in energy management programmes such as DR and can interconnect with the EMA including DERs, local generators (wind and solar) and energy storage devices.

NOTE The application message data and encoding for exchange can use the ISO/IEC 14543 series, ISO/IEC 10192-3, the IEC 62541 series and the ISO/IEC 30118 series.

The logical network arrangement of EMAs shall consist of a mesh or hierarchical tree structure. If a hierarchical tree topology is chosen for the interconnection among EMAs, the EMAP messages may be transferred without modification through an intermediate EMA. In addition, application layer messages as specified in the ISO/IEC 14543 series, ISO/IEC 10192-3, the IEC 62541 series and the ISO/IEC 30118 series may be passed across intermediate EMAs for service integration.

5.2 EMA protocol

The EMAP shall provide a logical interaction among EMAs. The EMAP enables energy management applications that co-operate and co-ordinate among EMAs. The EMAP specifies an application-layer protocol that includes protocol procedures and message formats specified in Clause 6. The EMAP specifies the protocol syntax, semantics, message formats, message sequences, etc. to ensure interoperability over a broad range of EMA deployments shown in Figure 2 to Figure 4. It also specifies a communication mechanism through which application messages may be passed among EMAs. Figure 2 shows a hierarchical EMAP application model in a residential area. Figure 3 shows a point-to-point EMAP application model. Figure 4 shows a hybrid EMAP application model. In these figures a product designed as an EMA can be an sEMA as well as a cEMA. In Figure 2, Figure 3 and Figure 4, arrows are logical connections and red arrows indicate HAN networks.

To support DR load control in a hierarchical situation, the sEMA shall send DR event or price signals to the cEMAs. If a cEMA accepts the DR event, then it subscribes to the DR event. Moreover, the cEMA adjusts the DR event within the available range to avoid penalties while performing the DR event. The cEMA can update the DR event if the cEMA is unable to limit the energy consumption under the subscribed control.

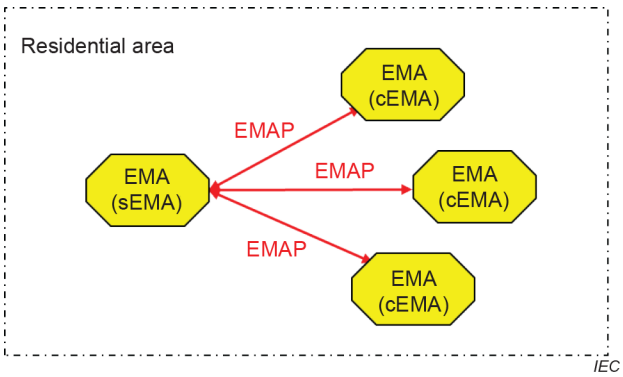


Figure 2 – A hierarchical EMAP application model in a residential area

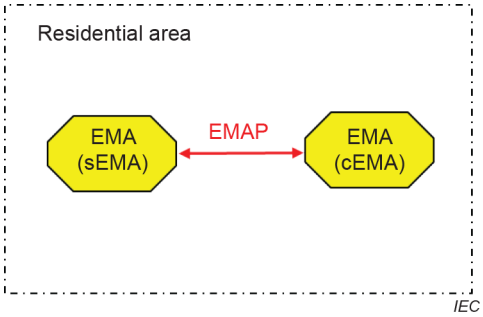


Figure 3 – A point-to-point EMAP application model in a residential area

This document specifies the protocol between EMAs in the point-to-point and hierarchical EMA configurations. Alternative configurations of an EMA framework architecture could be considered in a different standard.

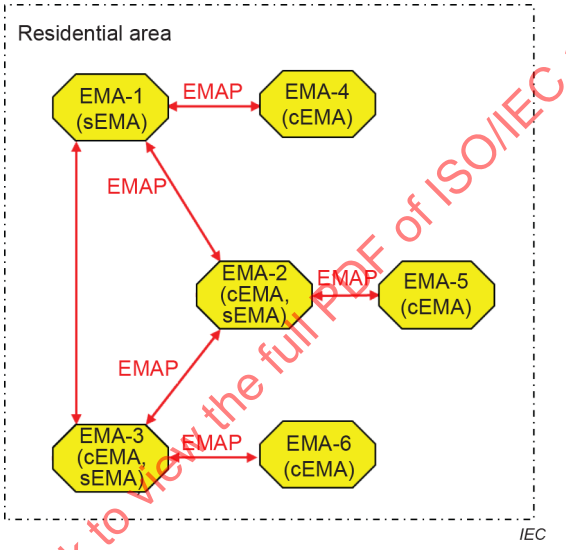


Figure 4 – A hybrid EMAP application model in a residential area

Clause 6 specifies a set of message exchanges for performing various functions and operations. Clause 7 contains a set of transport mechanisms for implementing the services. The transport mechanisms rely upon RFC 7252 "Constrained Application Protocol (CoAP)" and RFC 7159 "JavaScript object notation (JSON) Data Interchange format".

CoAP is a specialized Internet application protocol for devices with limited processing capability, as specified in RFC 7252. It enables EMA devices to communicate with the Internet using similar protocols. CoAP is designed for use between devices on the same constrained network (e.g. low-power wireless home networks), between devices and general nodes on the Internet, and between devices on different constrained networks both joined by an internet.

JSON is a public file format as specified in RFC 7159 that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format used for asynchronous browser–server communication.

Clause 6 specifies the following services:

1) RegisterParty: This service identifies a sEMA and a cEMA in advance of interactions using registration procedures.
2) Event: This service is used to call for service parameters and event information 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: The opt-in service and opt-out service enable the creation and termination of DR events generated by the service provider. The opt-change service overrides the availability schedule and addresses short-term changes in availability.

6 EMAP service procedure and message format

6.1 Overview

Clause 6 specifies protocol procedures and message formats to ensure interoperability among various implementations of EMAs. The application message set and data model (or schema) will be specified in other standards. Figure 5 illustrates a two step service procedure in the EMAP, which consists of the RegisterParty service and the event or report or opt service. The RegisterParty service identifies an sEMA and a cEMA in advance of interactions using registration procedures. After completion of party registration, the event or report or opt service is used to call for event information, the state of a resource, and changes in events, respectively.

In the service procedure the sEMA and cEMA shall communicate using CoAP, as defined in IETF RFC 7252, in either push mode (where the sEMA initiates communication) or in a pull mode (the cEMA requests information from the sEMA to begin a series of message exchanges).

The sequence diagrams shown in Clause 6 are for a push interaction pattern. In a pull mode the cEMA shall periodically poll the sEMA using Poll to provide the sEMA an opportunity to cancel the registration or request the cEMA to reregister. Refer to 6.6 for sample sequence diagrams of the pull interaction pattern.

These services use the CoAP connect process and subscription process between the sEMA and the cEMA. CoAP is a service layer protocol that is intended for use in resource-constrained internet devices, such as wireless sensor network nodes. CoAP is designed to translate easily to hypertext transfer protocol (HTTP) for simplified integration with the web, while also meeting specialized requirements such as multicast support, very low overhead, and simplicity.

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

All application-level error conditions shall be conveyed through the status code element of the CoAP response payload. Refer to 7.1.6 for error conditions.

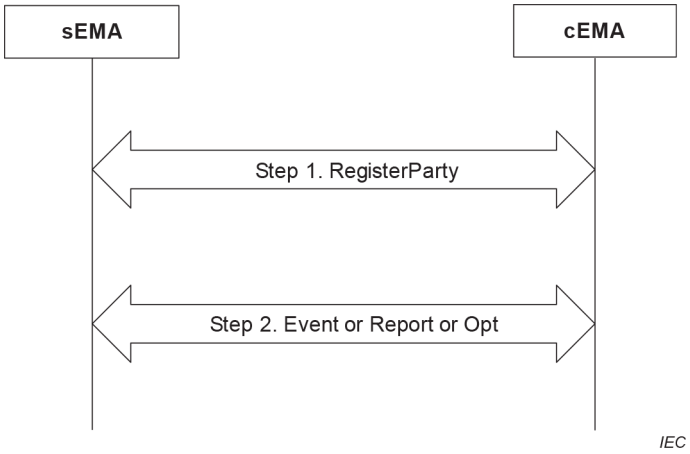


Figure 5 – Two-step service procedures of EMAP

The procedures shall employ JSON messaging over the CoAP transport mechanisms, with an example shown in Annex B. The message format is specified according to each protocol procedure between the sEMA and the cEMA in Clause 6.

Note that the application message sets used in this document are specified in the JSON schema file. The schema file will be specified in other standards.

The signal types and parameters for DERs are extended in both the event and report services.

6.2 RegisterParty

6.2.1 Overview

RegisterParty service uses in-band registration of cEMAs with sEMA. The following registration procedures are supported:

1) Query registration: a process to request registration information related to a DR event.
2) Create registration: a process to proceed with the actual registration to create the session for the communication based on the information from the query registration procedure.
3) Cancel registration: a process to terminate the registration that was created by the create registration procedure.
4) Request reregistration: a process to request registration again when registration information is lost under unexpected circumstances.

A cEMA initiates the query registration procedure to request registration information including all the profiles and transport names related to a DR event. Then, active registration is created by the cEMA with the create registration procedure. The sEMA or cEMA can cancel an active registration with the cancel registration procedure that was created from the create registration procedure. If the sEMA registration information changes, the sEMA can request registration again using request reregistration procedure when registration information is lost under unexpected circumstances.

6.2.2 Query registration

The query registration procedure begins with the cEMA requesting registration information related to a DR event (see 6.2.6.1) as shown in Figure 6. This query operation can be implemented using any of the supported transports. However, the cEMA shall be configured out-of-band with the address of the sEMA in order to initiate the query. The response to the query is the emapCreatedPartyRegistration (see 6.2.6.2). This payload contains information on all the profiles and transports supported by the sEMA in addition to any supported extensions to the profile. The information received by the cEMA can be used to determine the best configuration to use when formally registering as defined in the schema file.

NOTE Out-of-band configuration can be done by pre-programming or manual insertion during setup or by a discovery protocol.

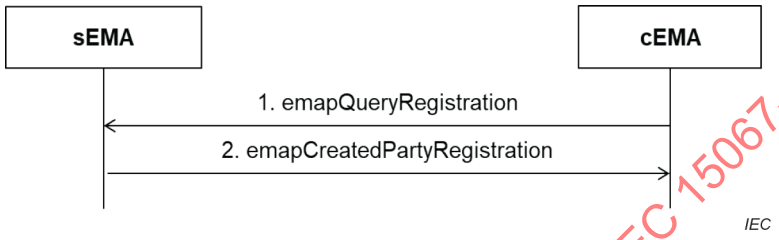


Figure 6 – Interaction diagram: Query registration

6.2.3 Create registration

Active registration shall be initiated by the cEMA with the emapCreatePartyRegistration (see 6.2.6.3) including registrationID as shown in Figure 7. The sEMA responds with an emapCreatedPartyRegistration containing all the profiles and transports supported by the sEMA, IDs, and other registration-related information (see 6.2.6.2). The registrationID in its response payload is used for subsequent operations pertaining to this registration instance. Based on the information including profiles and transport names, the cEMA has decided to create the session for communicating with the sEMA, in addition to other registration-related information.

If the cEMA registration information already exists, the cEMA can override the registration at any time using the emapCreatePartyRegistration payload referencing the current registrationID.

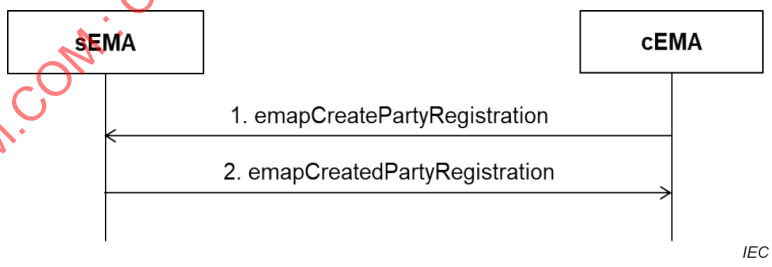


Figure 7 – Interaction diagram: Create registration

6.2.4 Cancel registration

The sEMA or cEMA shall cancel an active registration using the emapCancelPartyRegistration payload, referencing the registrationID. The other party responds with an emapCanceledPartyRegistration payload. This use case, depicted in Figure 8, shows how one party can perform cancel registration from the other party.

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

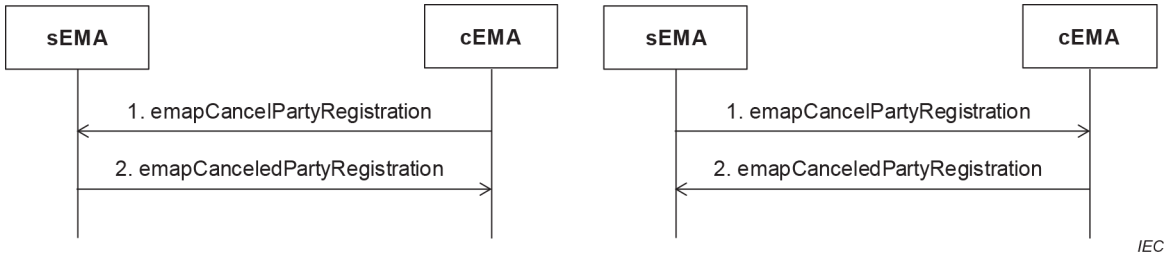


Figure 8 – Interaction diagram: Cancel registration

6.2.5 Request reregistration

If the sEMA registration information changes, the sEMA shall request registration again when registration information is lost under unexpected circumstances. The sEMA requests this change using the emapRequestReregistration (see 6.2.6.6) as shown in Figure 9. The response to this request is an emapResponse (see 6.2.6.7) as acknowledgement followed by an asynchronous create registration request from the cEMA.

If the cEMA registration information changes, the cEMA shall reregister the change at any time using the emapCreatePartyRegistration referencing the current registrationID as described in 6.2.3. The same registrationID will be maintained across re-registrations until one of the parties cancels the registration.

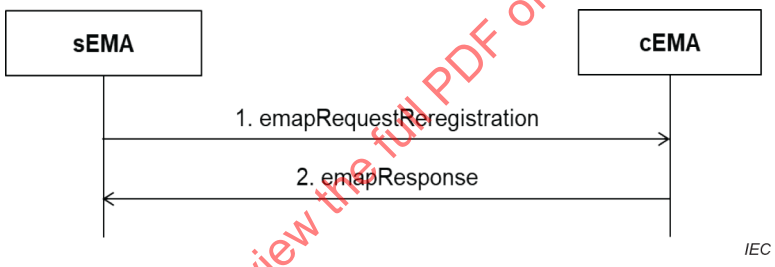


Figure 9 – Interaction diagram: Request reregistration – Push operation

6.2.6 Message formats for RegisterParty

6.2.6.1 emapQueryRegistration

Figure 10 describes the message format of emapQueryRegistration sent from the cEMA to the sEMA for requesting what profiles, transports, and extensions the sEMA may support. The emapQueryRegistration payload has the following components:

– schemaVersion: the version of EMAP data model schema used in this message;
– requestID: an identifier used to match up a logical transaction request and response.

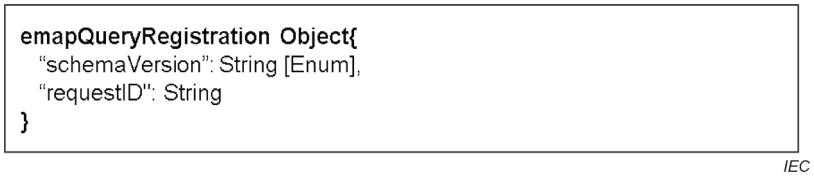


Figure 10 – emapQueryRegistration simplified message format

6.2.6.2 emapCreatedPartyRegistration

Figure 11 describes the message format of 6.2.6.2 emapCreatedPartyRegistration sent from the cEMA to the sEMA to report that the connection is successful. The emapCreatedPartyRegistration payload has the following components:

– schemaVersion: the version of EMAP data model schema used in this message;
– eiResponse: object that contains response status information whether received request is acceptable;
– registrationID: identifier for registration transaction;
– cEMAID: the ID of the cEMA;
– sEMAID: the ID of the sEMA;
– emapProfiles: profiles supported by the implementation;
– emapRequestedEmapPollFreq: object to identify polling frequency
– emapServiceSpecificInfo: service specific registration information;
– emapExtensions: an extension information for registration.

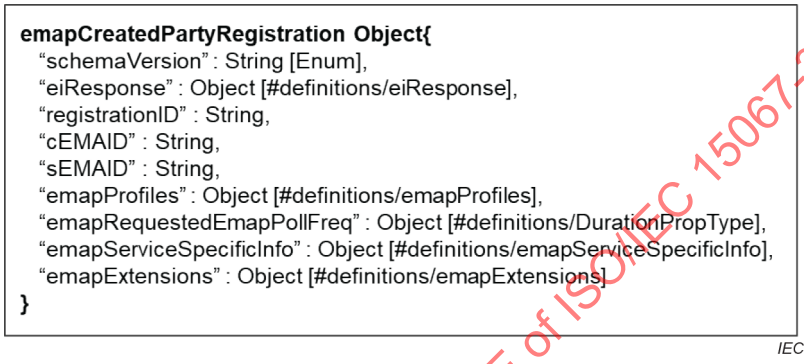


Figure 11 – emapCreatedPartyRegistration simplified message format

NOTE #definitions/eiResponse, #definitions/emapProfiles, #definitions/DurationPropType, #definitions/emapServiceSpecificInfo, #definitions/emapExtensions are defined in a schema file.

6.2.6.3 emapCreatePartyRegistration

Figure 12 describes the message format of emapCreatePartyRegistration sent from the cEMA to the sEMA to create the session for the communication.

The emapCreatePartyRegistration payload has the following components:

– schemaVersion: the version of schema used in this message;
– requestID: an identifier used to match up a logical transaction request and response;
– registrationID: an identifier for registration transaction;
– cEMAID: the ID of the cEMA;
– sEMAID: the ID of the sEMA;
– emapProfileName: the name of the profile to be used in the transaction;
– emapTransportName: the transport name, such as coap;
– emapTransportAddress: transport address used to communicate with other party;
– emapReportOnly: a flag value to indicate report only device;
– emapCEMAName: the name of cEMA;
– emapPullModel: Boolean to indicate whether the implementation uses the pull exchange model;
– optChangeable: Boolean to indicate the implementation uses the opt-change mode.

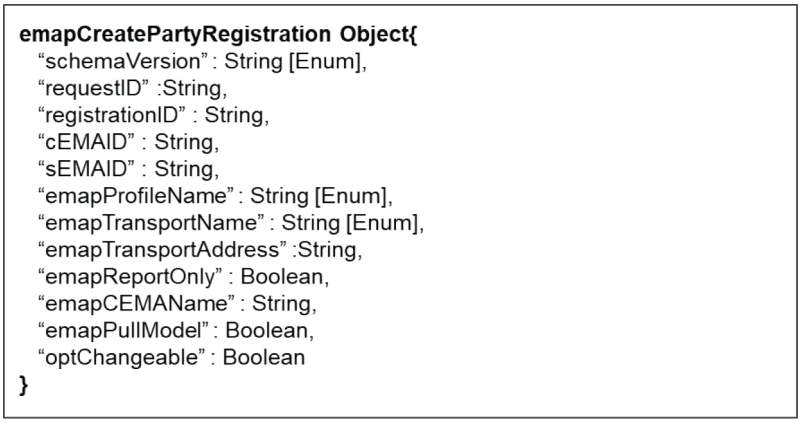


Figure 12 – emapCreatePartyRegistration simplified message format

6.2.6.4 emapCancelPartyRegistration

Figure 13 describes the message format of emapCancelPartyRegistration sent from the cEMA to the sEMA to cancel the registration. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– requestID: an identifier used to match up a logical transaction request and response;
– registrationID: an identifier for registration transaction;
– cEMAID: the ID of the cEMA.

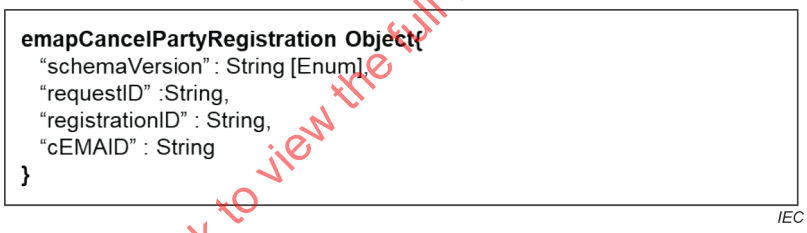


Figure 13 – emapCancelPartyRegistration simplified message format

6.2.6.5 emapCanceledPartyRegistration

Figure 14 describes the message format of emapCanceledPartyRegistration sent from the cEMA to the sEMA to cancel the registration. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– eiResponse: object that contains response status information whether received request is acceptable;
– registrationID: an identifier for registration transaction;
– cEMAID: the ID of the cEMA.

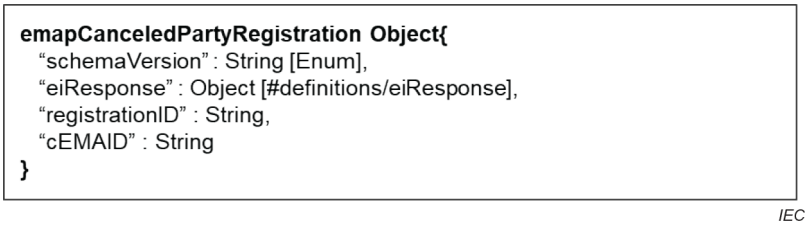


Figure 14 – emapCanceledPartyRegistration simplified message format

NOTE #definitions/eiResponse is defined in a schema file.

6.2.6.6 emapRequestReregistration

Figure 15 describes the message format sent from the cEMA to the sEMA to register the cEMA. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– cEMAID: the ID of the cEMA.

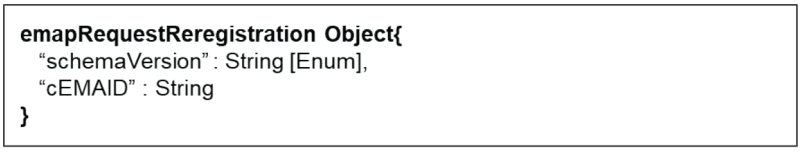


Figure 15 – emapRequestReregistration simplified message format

6.2.6.7 emapResponse

Figure 16 describes the message format of emapResponse sent from the sEMA to the cEMA to respond the registration. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– eiResponse: object that contains response status information whether received request is acceptable;
– cEMAID: the ID of the cEMA.

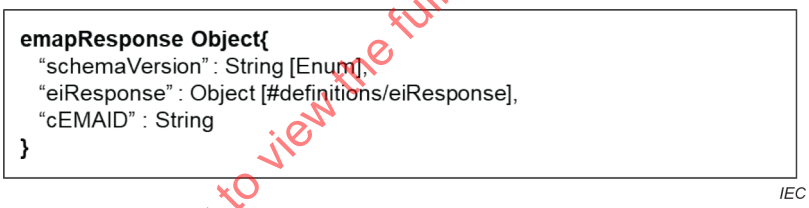


Figure 16 – emapResponse message format

NOTE #definitions/eiResponse is defined in a schema file.

6.3 Event

6.3.1 Overview

The event process uses the payloads below to subscribe events.

Events are generated by the sEMA and sent to the cEMA using the emapDistributeEvent. This payload contains all events applicable to the cEMA. Some events require a response and others do not, as indicated by the response Required element in the event description. If a response is required, the cEMA acknowledges its opt-in or opt-out disposition by responding with an emapCreatedEvent. This payload contains eventResponse elements matching each event. If no response is required, the cEMA shall not reply with emapCreatedEvent (or emapCreateOpt) payloads for this event.

When an event requires a response, an initial emapCreatedEvent is always sent from the cEMA to the sEMA. If a given programme allows a cEMA to later change its opt state during an event, it may do so by issuing a subsequent emapCreatedEvent (or emapCreateOpt) message containing the new state for a given event. Detailed descriptions of sEMA and cEMA event processing are given in 6.3.2 to 6.3.4.

Either a push or pull interaction operation may be used. For push, the sEMA shall deliver events to the cEMA using an emapDistributeEvent payload. In pull mode, the emapDistributeEvent shall be sent from the sEMA to the cEMA as response to a Poll (refer to 6.6). In addition to periodically sending Poll, a cEMA may also send one-time emapRequestEvent payloads to the sEMA in order to acquire events from the sEMA. If an application-level response is required, the cEMA asynchronously sends an emapCreatedEvent back to the sEMA in a second message.

The emapDistributeEvent payload contains additional signal type and values relevant to the DERs.

6.3.2 Push

Figure 17 illustrates push-based DR event subscription procedure, which has been issued by the event request using the emapDistributeEvent from the sEMA. The cEMA responds with CoAP 2.04. The CoAP 2.04 is a payload-free response to a CoAP PUT request indicating successful reception of the CoAP message according to the CoAP specification (RFC 7252). If an event response is required, the cEMA asynchronously acknowledges its opt-in or opt-out disposition by responding with an emapCreatedEvent to the sEMA. The sEMA responds with an emapResponse as acknowledgement of the emapCreatedEvent.

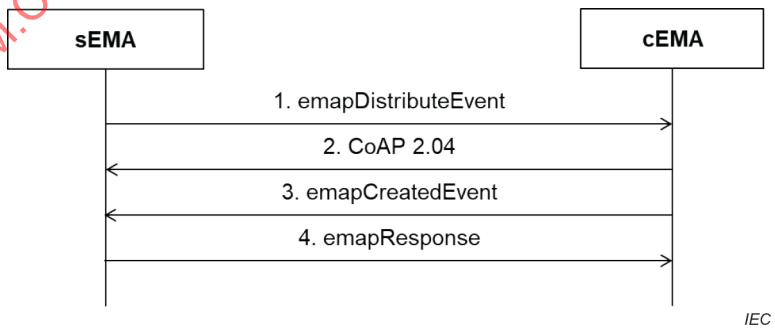


Figure 17 – Event interaction diagram – Push operation

6.3.3 Pull

Figure 18 illustrates pull-based DR event subscription procedure, which has been issued by the one-time request using emapRequestEvent or the periodic poll using emapPoll from the cEMA. sEMA responds to the event request using the emapDistributeEvent. If a response is required, the cEMA asynchronously acknowledges its opt-in or opt-out disposition by responding with an emapCreatedEvent to the sEMA. The sEMA responds with an emapResponse as acknowledgement.

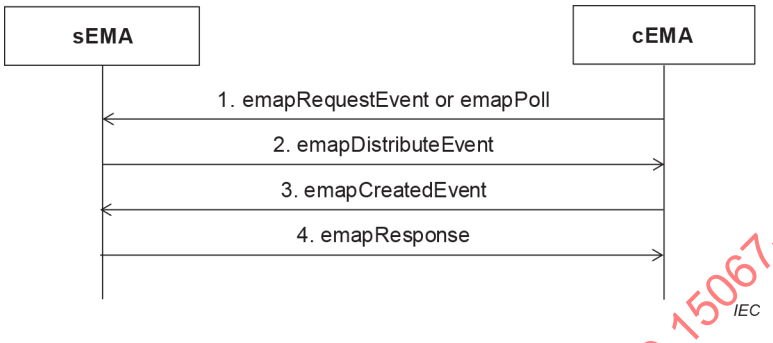


Figure 18 – Event interaction diagram – Pull operation

6.3.4 Message formats for event

6.3.4.1 General

For both push and pull operations, an emapDistributeEvent payload shall contain all events applicable to the cEMA it is communicating with.

In push mode, the cEMA's response to emapDistributeEvent is a transport level acknowledgement if required (2.04 response code).

6.3.4.2 emapRequestEvent

Figure 19 describes the RequestEvent message format sent from the cEMA to the sEMA in order to acquire events. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– eiRequestEvent: request event from a sEMA in pull mode.

```
emapRequestEvent Object{
  "schemaVersion" : String [Enum],
  "eiRequestEvent" : Object [#definitions/eiRequestEvent]
}
```

IEC

Figure 19 – emapRequestEvent simplified message format

NOTE #definitions/eiRequestEvent is defined in a schema file.

6.3.4.3 emapPoll

emapPoll is a service-independent polling mechanism used by cEMAs in a pull mode to request pending service operations to the sEMA. The cEMA queries the poll endpoint and the sEMA responds with the same message that it would have "pushed" had it been a push cEMA. If there are multiple messages pending a "push," the cEMA will continue to query the poll endpoint until there are no new messages and the sEMA responds with an emapResponse as acknowledgement. Figure 20 describes the message format of emapPoll message. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– cEMAID: the ID of the cEMA.

```
emapPoll Object{
  "schemaVersion" : String [Enum],
  "cEMAID" : String,
}
```

IEC

Figure 20 – emapPoll simplified message format

6.3.4.4 emapDistributeEvent

Figure 21 describes the emapDistributeEvent message format sent from the sEMA to the cEMA to distribute the DR event. Events are conveyed in the emapDistributeEvent payload using one or more event elements. The emapDistributeEvent payload has the following components:

– schemaVersion: the version of schema used in this message;
– eiResponse: object that contains response status information whether received request is acceptable;
– requestID: an identifier used to match up a logical transaction request and response;
– sEMAID: the ID of the sEMA;
– emapEvent: a set of objects containing a DR event.

The requestID uniquely identifies the request and any contained events. Its value is set by the sEMA using any scheme desired, including using the same value in every request if its use is not needed by the sEMA. The receiving cEMA shall use this requestID in the emapCreatedEvent event responses. The payload has the following components:

```
emapDistributeEvent Object{
  "schemaVersion" : String [Enum],
  "eiResponse" : Object [#definitions/eiResponse],
  "requestID" : String,
  "sEMAID" : String,
  "emapEvent" : Array [#definitions/emapEvent]
}
```

IEC

Figure 21 – emapDistributeEvent simplified message format

NOTE #definitions/eiResponse, #definitions/emapEvent are defined in a schema file.

6.3.4.5 emapCreatedEvent

Figure 22 describes the message format sent from the cEMA to the sEMA to report that the event was created based on the distributed DR event from the sEMA.

When one or more received events require a response, the cEMA creates and populates an emapCreatedEvent element and posts it to the sEMA. The "response" element contains an application level responseCode and responseDescription and a requestID. The optType may have a value of "optIn" or "optOut" to indicate the cEMAs disposition for a given event. An eventID contains a unique ID for this event. A modificationNumber contains a sequence that starts at zero and is incremented by one each time the sEMA modifies the event. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– eiCreatedEvent: a response to a DR event with optIn or optOut.

```
emapCreatedEvent Object{
  "schemaVersion" : String [Enum],
  "eiCreatedEvent" : Object [#definitions/eiCreatedEvent]
}
```

IEC

Figure 22 – emapCreatedEvent simplified message format

NOTE #definitions/eiCreatedEvent is defined in a schema file.

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

6.3.4.6 emapResponse

See 6.2.6.7.

6.4 Report

6.4.1 Overview

The reporting service supports the exchange of reports between the cEMA and the sEMA, and vice versa. All report interactions between the cEMA and the sEMA shall be built upon the following core operations:

– register report: a process to send its reporting capabilities;
– request report: a process to request a report that corresponds to its reporting capabilities;
– send report: a process to send a report with actual data elements;
– cancel report: a process to cancel ongoing (i.e. periodic) reports.

Figure 23 to Figure 26 show the logical payload exchanges that support each of these operations.

6.4.2 Register report

Register report process is performed after completion of party registration, both from the sEMA to the cEMA and from the cEMA to the sEMA. In addition, any party may send register report at any time after the initial registration.

In the push case, depicted in Figure 23, 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 interaction proceeds with the following steps.

1) The source party first sends its reporting capabilities to the target party by using the emapRegisterReport payload.
2) The target party responds with the emapRegisteredReport payload. The target party's response may contain an emapReportRequest 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 emapCreateReport as specified in 6.4.6.3.
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 emapCreatedReport payload.
4) The sEMA responds with an emapResponse as acknowledgement of the emapCreatedReport.

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 emapCreateReport, as specified in 6.4.6.3.

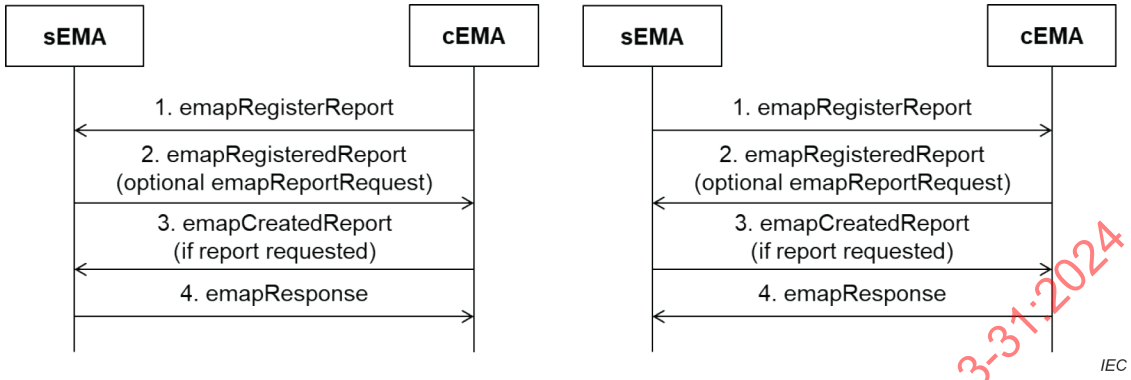


Figure 23 – Interaction diagram: Register report – Push operation

6.4.3 Request report

This use case, depicted in Figure 24, shows how one party can create reports from the other party.

The source party requests a report from the target party by using the emapCreateReport payload. That payload contains a set of reportSpecifierIDs that correspond to report capabilities in the report that was previously sent by the target party as part of the previously described emapRegisterReport interaction (refer to 6.4.2).

The response to the emapCreateReport payload is the emapCreatedReport payload.

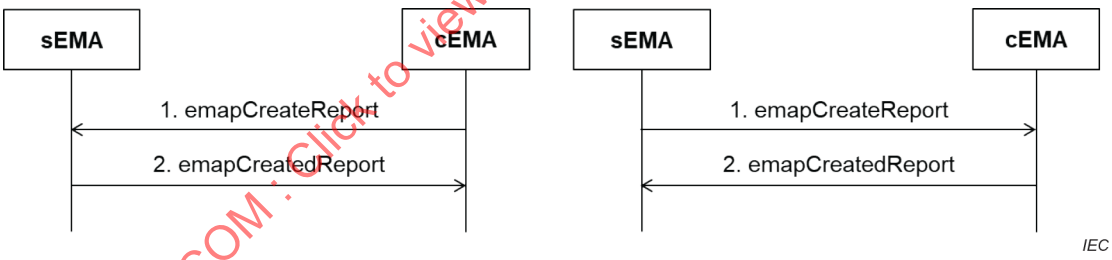


Figure 24 – Interaction diagram: Request report – Push operation

6.4.4 Send report

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

Figure 25 depicts the source party consistently sending a report to report the status to the target party. The source party sends emapUpdateReport to deliver the report. The target party replies with the emapUpdatedReport payload to acknowledge receipt of the report. As part of the emapUpdatedReport response, the target party may cancel the sending of any future reports using the optional emapCancelReport object, which contains a list of reportRequestIDs.

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 report object as the report registration operation did, but is used to exchange a report with actual data elements as opposed to the report used in the registration procedure.

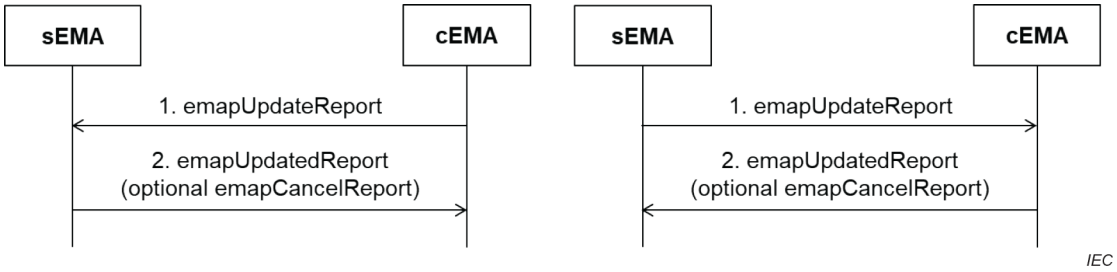


Figure 25 – Interaction diagram: Send report – Push operation

6.4.5 Cancel report

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

The source party sends emapCancelReport request to cancel ongoing (i.e. periodic) reports to the target party. The target party replies with emapCanceledReport as described in 6.4.6.9. This interaction depicted in Figure 26 is used by the source party to cancel ongoing (i.e. periodic) reports that are being generated by the target party.

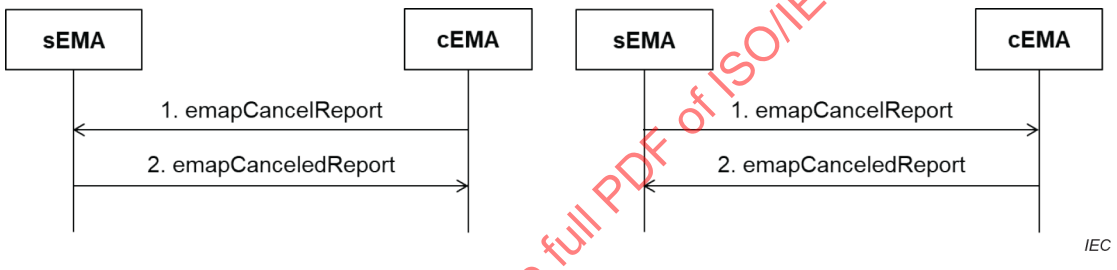


Figure 26 – Interaction diagram: Cancel report – Push operation

6.4.6 Message formats for report

6.4.6.1 emapRegisterReport

In general, the emapRegisterReport payload is the same as the emapUpdateReport payload except that it contains a report. The source party sends the special well-known report using the report schema as described in Figure 27.

As part of the report request, the source party specifies a set of reportRequestIDs that are used in subsequent operations on this report request. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– requestID: an identifier used to match up a logical transaction request and response;
– emapReport: a list of resources available for report;
– cEMAID: the ID of the cEMA;
– reportRequestID: an identifier of a particular report request.

```
emapRegisterReport Object{
  "schemaVersion" : String [Enum],
  "requestID" :String,
  "emapReport" : Array[#definitions/emapReport],
  "cEMAID" : String,
  "reportRequestID" : String
}
```

IEC

Figure 27 – emapRegisterReport simplified message format

NOTE #definitions/emapReport is defined in a schema file.

6.4.6.2 emapRegisteredReport

Figure 28 describes the emapRegisteredReport message format sent from the sEMA to the cEMA as notification that report registration was successful. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– eiResponse: object that contains response status information whether received request is acceptable;
– emapReportRequest: the object used for the report request;
– cEMAID: the ID of the cEMA.

```
emapRegisteredReport Object{
  "schemaVersion" : String [Enum],
  "eiResponse" : Object [#definitions/eiResponse],
  "emapReportRequest" : Array [#definitions/emapReportRequest],
  "cEMAID" : String
}
```

IEC

Figure 28 – emapRegisteredReport simplified message format

NOTE #definitions/eiResponse, #definitions/emapReportRequest are defined in a schema file.

The source party can only send the emapCreateReport after the target party has sent its METADATA report as part of the reporting registration procedure. 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.

6.4.6.3 emapCreateReport

Figure 29 describes the emapCreateReport message format sent from the cEMA to the sEMA to create report. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– requestID: an identifier used to match up a logical transaction request and response;
– emapReportRequest: object used for the report request;
– cEMAID: the ID of the cEMA.

```
emapCreateReport Object{
  "schemaVersion" : String [Enum],
  "requestID" : String,
  "emapReportRequest" : Array [#definitions/emapReportRequest],
  "cEMAID" : String
}
```

IEC

Figure 29 – emapCreateReport simplified message format

NOTE #definitions/emapReportRequest is defined in a schema file.

6.4.6.4 emapResponse

See 6.2.6.7.

6.4.6.5 emapCreatedReport

Figure 30 describes the message format sent from the sEMA to the cEMA as notification that report creation was successful. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– eiResponse: object that contains response status information whether received request is acceptable;
– emapPendingReports: the object contains a list of pending reports;
– cEMAID: the ID of the cEMA.

```
emapCreatedReport Object{
  "schemaVersion" : String [Enum],
  "eiResponse" : Object [#definitions/eiResponse],
  "emapPendingReports" : Object [#definitions/emapPendingReports],
  "cEMAID" : String
}
```

IEC

Figure 30 – emapCreatedReport simplified message format

NOTE #definitions/eiResponse, #definitions/emapPendingReports are defined in a schema file.

6.4.6.6 emapUpdateReport

Figure 31 describes the message format sent from the cEMA to the sEMA to report the status. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– requestID: an identifier used to match up a logical transaction request and response;
– emapReport: a recorded measurement stream on request;
– cEMAID: the ID of the cEMA.

```
emapUpdateReport Object{
  "schemaVersion" : String [Enum],
  "requestID" :String,
  "emapReport" : Array[#definitions/emapReport],
  "cEMAID" : String
}
```

IEC

Figure 31 – emapUpdateReport simplified message format

NOTE #definitions/emapReport is defined in a schema file.

6.4.6.7 emapUpdatedReport

Figure 32 describes the message format sent from the sEMA to the cEMA to report the status of the cEMA updated successfully. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– eiResponse: object that contains response status information whether received request is acceptable;
– emapCancelReport: information of the report to be cancelled;
– cEMAID: the ID of the cEMA.

```
emapUpdatedReport Object{
  "schemaVersion" : String [Enum],
  "eiResponse" : Object [#definitions/eiResponse],
  "emapCancelReport" :Object [#definitions/emapCancelReport],
  "cEMAID" : String
}
```

IEC

Figure 32 – emapUpdatedReport simplified message format

NOTE #definitions/eiResponse, #definitions/emapCancelReport are defined in a schema file.

6.4.6.8 emapCancelReport

Figure 33 describes the message format sent from the cEMA to the sEMA to cancel a report. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– requestID: an identifier used to match up a logical transaction request and response.
– reportRequestID: an identifier of a particular report request;
– reportToFollow: Boolean to indicate if report (in the form of UpdateReport) is to be returned following cancellation of report;
– cEMAID: the ID of the cEMA.

```
emapCancelReport Object{
  "schemaVersion" : String [Enum],
  "requestID" : String,
  "reportRequestID" : Array [#definitions/reportRequestID],
  "reportToFollow" : Boolean,
  "cEMAID" : String
}
```

IEC

Figure 33 – emapCancelReport simplified message format

NOTE #definitions/reportRequestID is defined in a schema file.

6.4.6.9 emapCanceledReport

Figure 34 describes the message format sent from the sEMA to the cEMA as notification that the report cancellation was successful. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– eiResponse: object that contains response status information whether received request is acceptable;
– emapPendingReports: the object contains a list of pending reports;
– cEMAID: the ID of the cEMA.

```
emapCanceledReport Object{
  "schemaVersion" : String [Enum],
  "eiResponse" : Object [#definitions/eiResponse],
  "emapPendingReports" : Object [#definitions/emapPendingReports],
  "cEMAID" : String
}
```

IEC

Figure 34 – emapCanceledReport simplified message format

NOTE #definitions/eiResponse, #definitions/emapPendingReports are defined in a schema file.

6.5 Opt

6.5.1 Overview

The opt service creates and communicates opt-in and opt-out schedules from the cEMA to the sEMA. These schedules specify temporary changes in the availability, and may be combined with longer term availability schedules and the marketContext requirements to give a complete picture of the willingness of the cEMA to respond to events received by the cEMA.

The cEMA uses the emapCreateOpt payload (see Figure 35) to accomplish one of the following objectives:

– to communicate to the sEMA a period of temporary availability for a specific set of targets,

– to communicate to the sEMA a period of temporary un-availability for a specific set of targets,

– to optIn, optOut or optChange of a previously acknowledged event for a specific set of targets.

6.5.2 Create opt

The emapCreateOpt payload includes an optID, which can be used in subsequent operations to reference this schedule. The sEMA's response to emapCreateOpt is an emapCreatedOpt payload that includes an optID, which can be used in subsequent operations to reference this schedule.

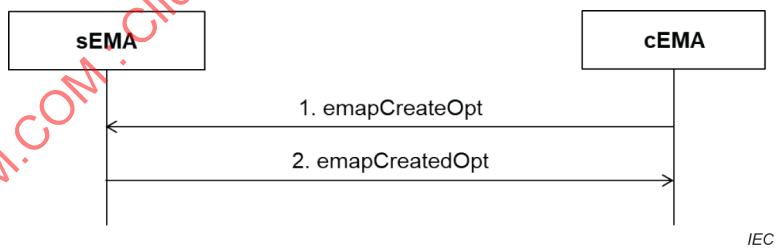


Figure 35 – Interaction diagram: Create opt

6.5.3 Cancel opt

The cEMA may at any time cancel a temporary availability schedule by using the emapCancelOpt with an optID referencing the schedule to be cancelled (see Figure 36). The sEMA sends emapCanceledOpt including an optID from sEMA to cEMA to respond to the emapCancelOpt.

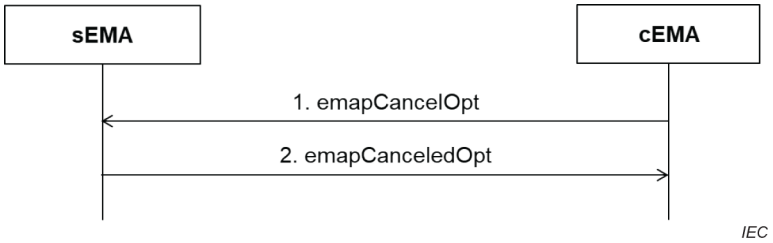


Figure 36 – Interaction diagram: Cancel opt

6.5.4 Message formats for opt

6.5.4.1 emapCreateOpt

Figure 37 describes the message format sent from the cEMA to the sEMA to report available schedule and opt state changes. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– optID: an identifier for an opt interaction;
– optType: an indicator to identify the selected opt mode (opt-in, opt-out or opt-change);
– optValue: object to identify the proposed value to change;
– optReason: an enumerated value for the opt reason;
– marketContext: a URI identifying a DR programme;
– cEMAID: the ID of the cEMA;
– vavailability: a value of schedule reflecting device availability for participating in DR events;
– createdDateTime: the dateTime the payload was created;
– requestID: an identifier used to match up a logical transaction request and response;
– qualifiedEventID: a fully qualified event ID;
– eiTarget: an object that identifies the resources associated with the logical cEMA interface;
– emapDeviceClass: a Device Class target.

A cEMA shall request a change to an ongoing event by setting the optType value to optChange, with an example shown in Annex A. The proposed value is passed to the sEMA via optValue, where the response to the proposal is communicated via the emapCreatedOpt message.


```
emapCreateOpt Object{
  “schemaVersion” : String [Enum],
  “optID” : String,
  “optType” : String [Enum], //opt in, opt out, opt Change
  “optValue” : Object [#definitions/currentValueType],
  “optReason” : String [Enum],
  “marketContext” : String,
  “cEMAID” : String,
  “vavailability” : Object [#definitions/vavailability],
  “createdDateTime” : String,
  “requestID” : String,
  “qualifiedEventID” : Object [#definitions/qualifiedEventID],
  “eiTarget” : Object [#definitions/eiTarget],
  “emapDeviceClass” : Object [#definitions/EiTargetType]
}
```

IEC

Figure 37 – emapCreateOpt simplified message format

NOTE #definitions/currentValueType, #definitions/vavailability, #definitions/qualifiedEventID, #definitions/eiTarget, #definitions/EiTargetType are defined in a schema file.

6.5.4.2 emapCreatedOpt

Figure 38 describes the message format sent from the sEMA to the cEMA in response to the emapCreateOpt. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– eiResponse: object that contains response status information whether received request is acceptable;
– optID: an identifier for an opt interaction.

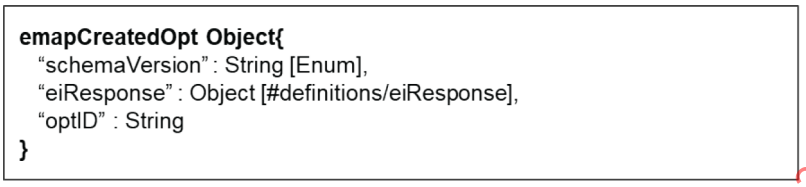


Figure 38 – emapCreatedOpt simplified message format

NOTE #definitions/eiResponse is defined in a schema file.

6.5.4.3 emapCancelOpt

Figure 39 describes the message format sent from the cEMA to the sEMA to cancel the emapCancelOpt. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– requestID: an identifier used to match up a logical transaction request and response;
– optID: an identifier for an opt interaction;
– cEMAID: the ID of the cEMA.

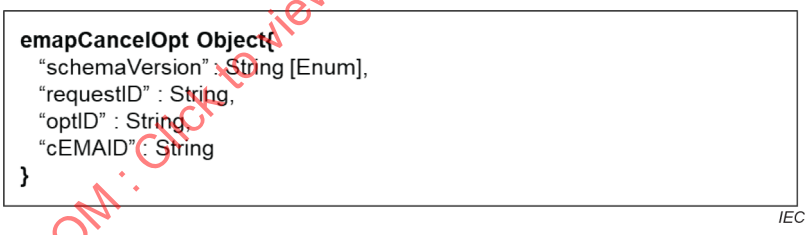


Figure 39 – emapCancelOpt simplified message format

6.5.4.4 emapCanceledOpt

Figure 40 describes the message format sent from the cEMA to the sEMA to respond to the emapCanceledOpt. The payload has the following components:

– schemaVersion: the version of schema used in this message;
– eiResponse: object that contains response status information whether received request is acceptable;
– optID: an identifier for an opt interaction.

```
emapCanceledOpt Object{
  "schemaVersion" : String [Enum],
  "eiResponse" :Object [#definitions/eiResponse],
  "optID" : String
}
```

IEC

Figure 40 – emapCanceledOpt simplified message format

NOTE #definitions/eiResponse is defined in a schema file.

6.6 Poll

As new services are added, there is a need for pull-only cEMAs to poll the sEMA periodically when a sEMA might want to send some information, particularly for sEMA-initiated use cases where the information is not necessarily periodic and the cEMA cannot predict. In the push mode, this is a non-issue because the sEMA can initiate the operations. However, in the pull mode, this requires the cEMA to poll the sEMA periodically, possibly at multiple different service endpoints, to retrieve all the information that the sEMA might want to provide. Poll provides a solution that allows the pull cEMA to emulate the push message exchange pattern from sEMA to cEMA.

Poll is a service-independent polling mechanism used by cEMAs in a pull mode to request pending service operations to the sEMA. The cEMA queries the poll endpoint and the sEMA responds with the same message that it would have pushed had it been a push cEMA. If there are multiple messages pending a push, the cEMA will continue to query the poll endpoint until there are no new messages and the sEMA responds with a response payload.

The allowed response payloads sent by the sEMA in response to an emapPoll request are shown in Figure 41 to Figure 48:

– emapResponse (see Figure 41);
– emapCancelPartyRegistration (see Figure 42);
– emapRequestReregistration (see Figure 43);
– emapDistributeEvent (see Figure 44 and 6.3.3);
– emapRegisterReport (see Figure 45);
– emapCreateReport (see Figure 46);
– emapUpdateReport (see Figure 47);
– emapCancelReport (see Figure 48).

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

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

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

The sequence diagrams in Figure 41 to Figure 48 illustrate typical patterns using pull-based procedures. The cEMA queries the poll endpoint and the sEMA responds with the same message.

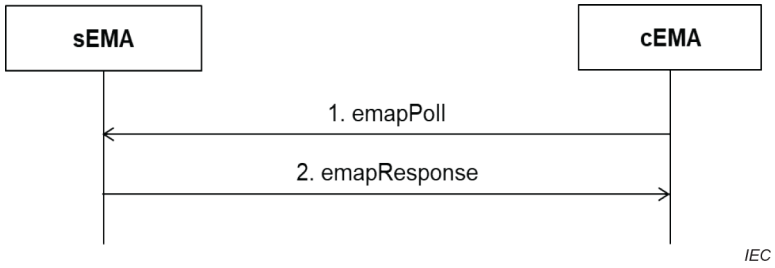


Figure 41 – Interaction diagram: Poll (nothing in queue)

Figure 41 illustrates that the sEMA receives an emapPoll message and responds with an emapResponse message when there are no more messages to send.

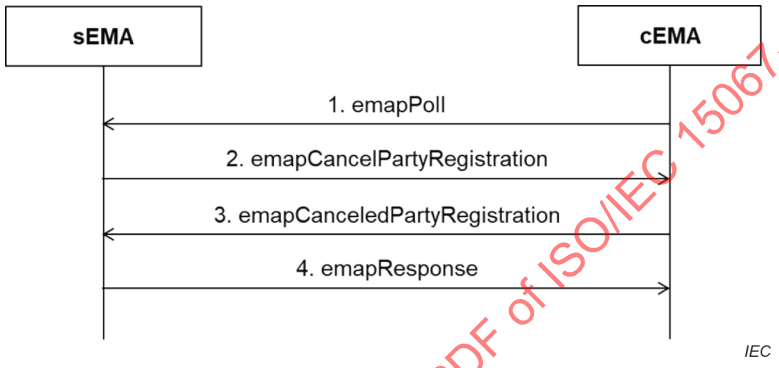


Figure 42 – Interaction diagram: Poll (emapCancelPartyRegistration reply)

Figure 42 illustrates pull-based cancel registration procedure, which describes how one party can cancel an active registration. This has been issued by the periodic poll using emapPoll from the cEMA. sEMA responds to the request using the emapCancelPartyRegistration. Then the source party responds with the emapCanceledPartyRegistration payload. The sEMA responds with an emapResponse as acknowledgement of the emapCanceledPartyRegistration.

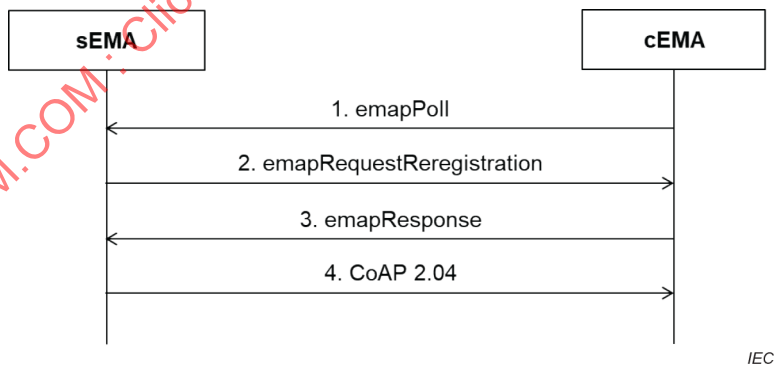


Figure 43 – Interaction diagram: Poll (emapRequestReregistration reply)

Figure 43 illustrates pull-based request reregistration procedure, which describes how an sEMA can request reregistration. This has been issued by the periodic poll using emapPoll from the cEMA. sEMA responds to the request using the emapRequestReregistration. Then the source party responds with the emapResponse payload. The sEMA responds with a CoAP 2.04 as acknowledgement of the emapResponse.

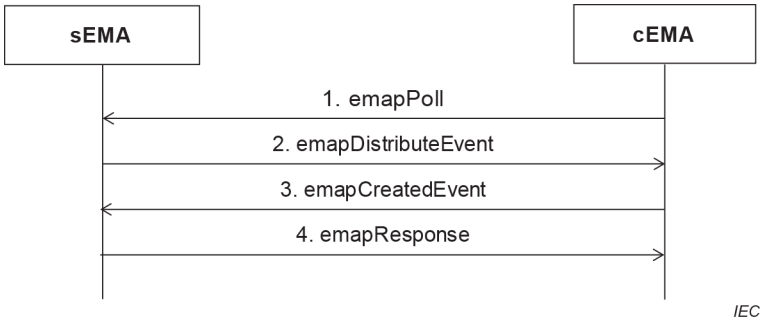


Figure 44 – Interaction diagram: Poll (emapDistributeEvent reply)

Figure 44 illustrates pull-based DR event subscription procedure, which has been issued by the periodic poll using emapPoll from the cEMA. sEMA responds to the event request using the emapDistributeEvent. Then the source party responds with the emapCreatedEvent payload. The sEMA responds with an emapResponse as acknowledgement of the emapCreatedEvent.

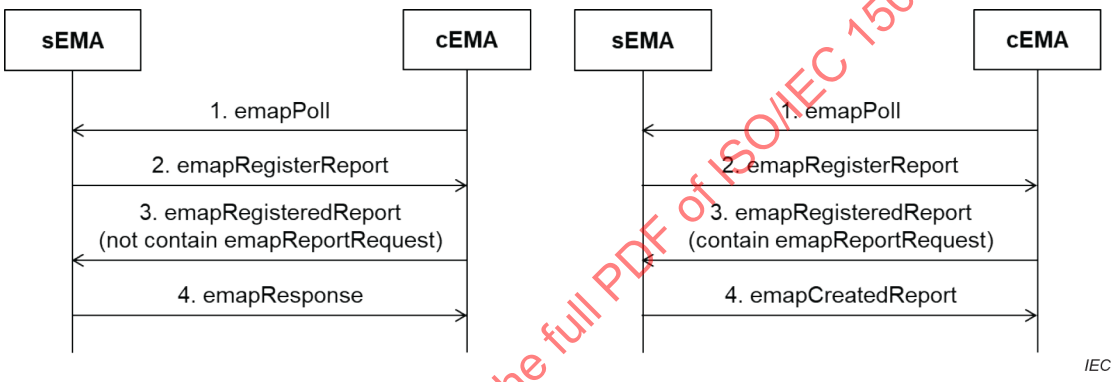


Figure 45 – Interaction diagram: Poll (emapRegisterReport reply)

Figure 45 illustrates pull-based register report procedure, which describes how one party can register a report from the other party. This has been issued by the periodic poll using emapPoll from the cEMA. sEMA responds to the request using the emapRegisterReport. Then the source party responds with the emapRegisteredReport payload. If the emapReportRequest contains an emapRegisteredReport message, the sEMA responds with an emapCreatedReport. If the emapRegisteredReport message does not contain an emapReportRequest, the sEMA responds with an emapResponse.

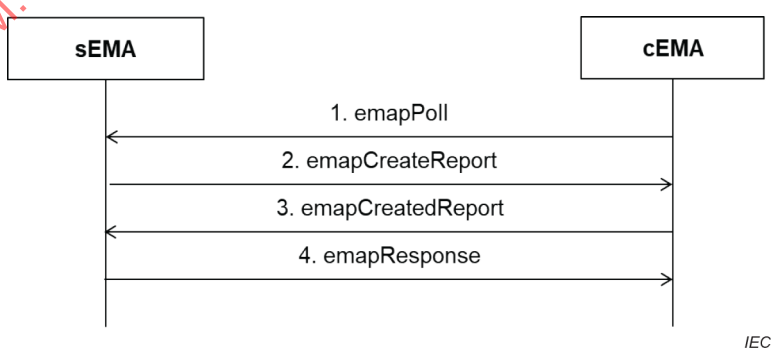


Figure 46 – Interaction diagram: Poll (emapCreateReport reply)

Figure 46 illustrates pull-based emapCreateReport procedure, which describes how one party can create report from the other party. This has been issued by the periodic poll using emapPoll from the cEMA. sEMA responds to the event request using the emapCreateReport. Then the source party responds with the emapCreatedReport payload. The sEMA responds with an emapResponse as acknowledgement of the emapCreatedReport.

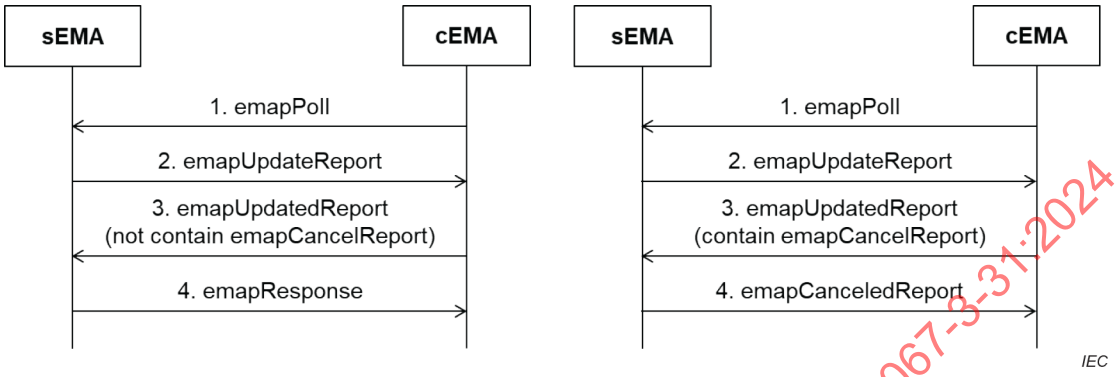


Figure 47 – Interaction diagram: Poll (emapUpdateReport reply)

Figure 47 illustrates pull-based send report procedure, which describes how one party can receive periodic reports from the other party. This has been issued by the periodic poll using emapPoll from the cEMA. An sEMA responds to the request with the emapUpdateReport. Then the source party responds with the emapUpdatedReport payload. If the emapUpdatedReport message contains an emapCancelReport, the sEMA responds with an emapCanceledReport. If the emapUpdatedReport message does not contain an emapCancelReport, the sEMA responds with an emapResponse.

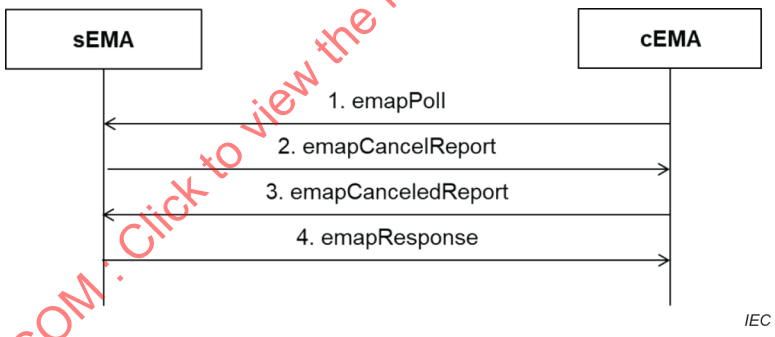


Figure 48 – Interaction diagram: Poll (emapCancelReport reply)

Figure 48 illustrates pull-based cancel report procedure, which describes how one party can cancel periodic reports from the other party. This has been issued by the periodic poll using emapPoll from the cEMA. sEMA responds to the request using the emapCancelReport. Then the source party responds with the emapCanceledReport payload. The sEMA responds with an emapResponse as acknowledgement of the emapCanceledReport.

7 Transport protocol

7.1 CoAP

7.1.1 General

CoAP in this document refers to a CoAP implementation that uses CoAP PUT to propagate the payloads. Each request is carried in a confirmable (CON) message. If the server is able to process the request (from a CON message) immediately, the request can be acknowledged by the response within the Acknowledgement (ACK) message. Otherwise, it sends an empty ACK with the response codes (CoAP status). If the server receives a CON message but is not able to process it, a Reset message (RST) is transmitted back to the client, instead of an ACK.

The CoAP layer specifies the messaging, including the four types of request methods (GET, PUT, POST and DELETE) and the response codes (Success 2.xx, Client Error 4.xx and Server Error 5.xx).

The CoAP servers allow the discovery feature to provide services by exposing the URIs of resources in the namespace of the servers. Resource discovery queries must always be made directly to the servers that offer services. This can be done directly if a client knows the default server URI, or alternatively by using multicast CoAP addresses to find all CoAP servers on a network. The discovery service should be offered through a default known port (5683, or 5684 for secured communication).

7.1.2 Push and pull implementation

7.1.2.1 Push definition

In push mode, messages may be sent from the sEMA to the cEMA (pushed). In order to use push, the cEMA shall expose CoAP URI endpoints (a CoAP server) to which the sEMA may send requests such as DistributeEvent. While this is the most efficient way to execute the communication over CoAP, it presents technical difficulties (e.g. port forwarding) as the cEMA can reside behind a gateway.

7.1.2.2 Pull definition

In pull mode, all operations are initiated by the cEMA to the sEMA. This can be thought of as a "polling" mode, where the cEMA periodically asks for updates to the sEMA. The pull mode removes the requirement for a CoAP server on the cEMA, avoiding the technical limitation presented by the "Pass-Through" mode of a gateway in front of a cEMA. However, the pull mode has 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 cEMA initiating the CoAP request. In a push mode, the sEMA can notify a cEMA of a new event via the DistributeEvent operation. The sEMA would send a request with an emapDistributeEvent payload, to which the cEMA responds with CoAP 2.04 followed by an asynchronous CreatedEvent.

However, in the pull mode, the cEMA requests events from the sEMA using Poll, to which a emapDistributeEvent payload is sent in the response. After parsing the response, the cEMA still needs to acknowledge the creation of any new events by making a second request using the CreatedEvent operation on the cEMA.

7.1.3 Service endpoint URIs

The endpoint address will be of the URI form:

coap://<hostname>(:port)/(prefix/)EMAP/1.0/<service>

– "hostname" is the endpoint name or IP address and "port" is an optional.
– "prefix" is an optional URI path prefix that can be used to separate EMAP services from other services that can reside on the same CoAP server.
– "service" is the name of the service (e.g. "emapRequestEvent", "emapCreateReport", "emapCancelOpt", "emapQueryRegistration", "emapPoll").

The "service" portion is defined by the JSON payload sent in a request.

The information from cEMA to sEMA should flow through the service endpoints. The same well-defined service endpoints are used by CoAP when communicating in the opposite direction. The emapCreatePartyRegistraion:TransportAddress element should contain the IP address (e.g. 129.6.252.49) for the cEMA, such as "coap://129.6.252.49:5683/prefix/". When sending information to the cEMA, the sEMA should concatenate the IP address to the well-known service endpoints to form a complete endpoint address such as coap://129.6.252.49:5683/prefix/EMAP/1.0/emapRegisterReport.

Implementations that expose both a cEMA and sEMA (such as an aggregator) shall use different URI endpoints for the sEMA and cEMA interfaces, for example:

- coap://mycompany.com/mysEMA/EMAP/1.0/emapRegisterReport
- coap://mycompany.com/mycEMA/EMAP/1.0/emapRegisterReport

The "service" portion of an endpoint URI can be specified by the JSON payload sent in a request, for example, an emapRegisterReport payload specifies the register report operation.

7.1.4 CoAP methods

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

7.1.5 Failure conditions

The following failures can occur for a given operation:

– UDP (or below) fails;
– Datagram Transport Layer Security (DTLS) negotiation fails;
– CoAP fails (CoAP 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 UDP failure, it is always recommended to retry the operation.

7.1.6 CoAP response codes

The following CoAP status codes, defined in IETF RFC 7252, are used in this document:

2.04 Changed – 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 CoAP error codes.

4.04 Not Found – the cEMA or sEMA does not support the requested operation. The requestor shall not re-send the request.

4.06 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.

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

5.03 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 pause the procedure in order to not put further strain on the server. The requestor shall retry the request after the proper preparation period.

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

For all error (non-2.04) 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 response payload.

8 Security

All CoAP messages shall be sent as datagram transport layer security (DTLS) [IETF RFC 6347] "application data". For matching an ACK or Reset message (RST) to a confirmable message: For matching a response to a request, the DTLS session shall be the same and the epoch shall be the same. The response to a DTLS secured request shall always be DTLS secured using the same security session and epoch.

DTLS defines four security options: none (NoSec); pre-shared keys that determine authorized peer-to-peer communication at the node level (PreSharedKey); public keys that authorize communication to a set of nodes (RawPublicKey); and certificates that can be used instead of public keys to sign message content (Certificate).