

DIN EN 14908-10

ICS 35.240.67; 91.140.01; 97.120

Einsprüche bis 2024-03-19

Entwurf

**Firmenneutrale Datenkommunikation für die Gebäudeautomation und Gebäudemanagement –
Gebäude-Netzwerk-Protokoll –
Teil 10: Spezifikation der Webdienste für das
Kontrollnetzwerkprotokoll;
Englische Fassung prEN 14908-10:2023**

Open Data Communication in Building Automation, Controls and Building Management –
Control Network Protocol –
Part 10: Web Services for Control Networking Protocol Specification;
English version prEN 14908-10:2023

Réseau ouvert de communication de données pour l'automatisation, la régulation et la gestion
technique du bâtiment –
Protocole de contrôle du réseau –
Partie 10: Services Web pour la spécification du protocole de réseau de contrôle;
Version anglaise prEN 14908-10:2023

Anwendungswarnvermerk

Dieser Entwurf mit Erscheinungsdatum 2024-01-19 wird der Öffentlichkeit zur Prüfung und Stellungnahme vorgelegt.

Weil das beabsichtigte Dokument von der vorliegenden Fassung abweichen kann, ist die Anwendung dieses Entwurfs besonders zu vereinbaren.

Stellungnahmen werden erbeten

- vorzugsweise online im Norm-Entwurfs-Portal von DIN unter www.din.de/go/entwuerfe bzw. für Norm-Entwürfe der DKE auch im Norm-Entwurfs-Portal der DKE unter www.entwuerfe.normenbibliothek.de, sofern dort wiedergegeben;
- oder als Datei per E-Mail an nhrs@din.de möglichst in Form einer Tabelle. Die Vorlage dieser Tabelle kann im Internet unter www.din.de/go/stellungnahmen-norm-entwuerfe oder für Stellungnahmen zu Norm-Entwürfen der DKE unter www.dke.de/stellungnahme abgerufen werden;
- oder in Papierform an den DIN-Normenausschuss Heiz- und Raumluftechnik sowie deren Sicherheit (NHRs), 10772 Berlin oder Am DIN-Platz, Burggrafenstr. 6, 10787 Berlin.

Es wird gebeten, mit den Kommentaren zu diesem Entwurf jegliche relevanten Patentrechte, die bekannt sind, mitzuteilen und unterstützende Dokumentationen zur Verfügung zu stellen.

Gesamtumfang 456 Seiten

DIN-Normenausschuss Heiz- und Raumluftechnik sowie deren Sicherheit (NHRs)

Nationales Vorwort

Dieses Dokument (prEN 14908-10:2023) wurde vom Technischen Komitee CEN/TC 247 „Gebäudeautomation und Gebäudemanagement“ erarbeitet, dessen Sekretariat von SNV (Schweiz) gehalten wird.

Das zuständige deutsche Normungsgremium ist der Arbeitsausschuss NA 041-03-65 AA „Gebäudeautomation: Produkte, Systeme und Kommunikation (SpA CEN/TC 247 und ISO/TC 205/WG 3)“ im DIN-Normenausschuss Heiz- und Raumlufttechnik sowie deren Sicherheit (NHRS).

Dieses Dokument enthält unter Berücksichtigung des DIN-Präsidialbeschlusses 1/2004 nur die Englische Originalfassung von prEN 14908-10:2023.

Aktuelle Informationen zu diesem Dokument können über die Internetseiten von DIN (www.din.de) durch eine Suche nach der Dokumentennummer aufgerufen werden.

Nationaler Anhang NA (informativ)

Begriffe

Reihenfolge und Inhalt der folgenden Begriffe sind identisch mit denen im Abschnitt Begriffe der englischen Fassung.

Für die Anwendung dieses Dokuments gelten die Begriffe nach ISO 21549-1 und die folgenden Begriffe.

ISO und IEC stellen terminologische Datenbanken für die Verwendung in der Normung unter den folgenden Adressen bereit:

- ISO Online Browsing Platform: verfügbar unter <https://www.iso.org/obp>
- IEC Electropedia: verfügbar unter <https://www.electropedia.org/>

3.1

Message Queuing Telemetry Transport MQTT

Leichtes Publish/Subscribe-, Maschine-zu-Maschine-Netzwerkprotokoll für Nachrichtenwarteschlangen-/Message Queuing-Dienstleistungen

3.2

REpresentational State Transfer REST

Software-Architektur-Typ, der eine einheitliche Schnittstelle zwischen physisch getrennten Komponenten, häufig über das Internet in einer Client-Server-Architektur, beschreibt

3.3

Internet Access Protocol IAP

offenes und erweiterbares Protokoll für das Internet of Things zur Bereitstellung und Verwaltung von Geräten und zum Zugriff auf und Aktualisierung von Daten der Geräte

— Leerseite —

Open Data Communication in Building Automation,
Controls and Building Management - Control Network
Protocol - Part 10: Web Services for Control Networking
Protocol Specification

Firmenneutrale Datenkommunikation für die
Gebäudeautomation und Gebäudemanagement -
Gebäude-Netzwerk-Protokoll - Teil 10: Spezifikation
der Webdienste für das Kontrollnetzwerkprotokoll

This draft European Standard is submitted to CEN members for enquiry. It has been drawn up by the Technical Committee CEN/TC 247.

If this draft becomes a European Standard, CEN members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

This draft European Standard was established by CEN in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CEN member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CEN members are the national standards bodies of Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Republic of North Macedonia, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland, Türkiye and United Kingdom.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Warning : This document is not a European Standard. It is distributed for review and comments. It is subject to change without notice and shall not be referred to as a European Standard.



EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

CEN-CENELEC Management Centre: Rue de la Science 23, B-1040 Brussels

Contents	Page
European foreword	9
Introduction	10
1 Scope.....	12
2 Normative references.....	12
3 Terms and definitions	12
4 IAP API FUNDAMENTALS.....	13
4.1 IAP API Overview	13
4.2 IAP/MQ Fundamentals.....	14
4.2.1 IAP/MQ Overview	14
4.2.2 MQTT in IAP	14
4.2.3 Supported MQTT Version	14
4.2.4 IAP/MQ Topics	15
4.2.5 Persistent Clients vs Transient Clients	15
4.2.6 Client Expiration	15
4.2.7 Message Payload	15
4.2.8 CMS Connectivity	16
4.2.9 An IAP Site	16
4.3 Partial Object Assignment	16
4.3.1 Definition	16
4.3.2 Edge Servers and Partial Object Assignment.....	17
4.3.3 Timestamps in IAP	17
4.3.4 Most Recently Used (MRU) Timestamp	18
4.4 IAP/MQ Topic Syntax.....	18
4.4.1 Syntax elements.....	18
4.4.2 Device Syntax.....	23
4.4.3 Wildcards.....	25
4.5 IAP/REST Fundamentals	25
4.5.1 REST Overview	25
4.5.2 REST in IAP	26
4.5.3 Making IAP/REST Calls	26
4.6 IAP/REST Syntax	26
4.7 Queries and Parameters.....	28
4.8 Path Parameters.....	30
4.9 Query Parameters.....	31
4.10 IAP/WS Fundamentals	32
4.11 Data Log Websocket Queries and Parameters	33
4.11.1 Data Log Websocket Overview.....	33
4.11.2 Key.....	33
4.11.3 Operator	33
4.11.4 Value	33
4.11.5 Regular Expression Encoding.....	33
5 IAP/MQ API	34
5.1 Overview	34
5.2 About	35
5.2.1 Definition	35

5.2.2	Example.....	35
5.2.3	Properties.....	36
5.3	Alarm.....	40
5.3.1	Definition.....	40
5.3.2	Example.....	41
5.3.3	Alarm Configuration.....	42
5.3.4	Alarm Status	42
5.3.5	Examples.....	43
5.3.6	Alarm Properties	43
5.3.7	Alarm Actions.....	44
5.3.8	Alarm Conditions.....	46
5.3.9	Manage Alarms	48
5.4	Connections	49
5.4.1	Definition.....	49
5.4.2	Connection Implementation.....	49
5.4.3	Connection with Datapoint Presets	50
5.4.4	Connection Objects.....	50
5.4.5	Connection Do Actions.....	56
5.4.6	Connection Types	59
5.4.7	Type Translation.....	60
5.5	Data Type Definitions.....	64
5.5.1	Definition.....	64
5.5.2	Data Type References.....	64
5.5.3	About IAP Data Types.....	65
5.5.4	Data Type General Information.....	66
5.5.5	Data Type Presentation and Formatting.....	67
5.6	Device.....	68
5.6.1	Definition.....	68
5.6.2	Device Objects.....	69
5.6.3	Device Configuration.....	70
5.6.4	Device Do Action	71
5.6.5	Device Interface	82
5.6.6	Device Status	82
5.6.7	Dynamic Interfaces	84
5.6.8	Implementation Detail Object.....	86
5.7	Discovery	88
5.8	Discovery and Provisioning Details.....	90
5.8.1	Device Discovery.....	90
5.8.2	Segment Discovery.....	94
5.8.3	Segment Provisioning	95
5.8.4	Segment Discovery Message.....	97
5.9	Events.....	100
5.9.1	Definition.....	100
5.9.2	Data Events	101
5.9.3	Join Events.....	101
5.9.4	Tracing.....	102
5.9.5	Event Object Properties	102
5.9.6	Created Event.....	105
5.9.7	Deleted Event.....	107
5.9.8	Updated Event.....	108
5.10	Group.....	109
5.10.1	Definition.....	109
5.10.2	Group Characteristics	110

5.10.3	Group Examples.....	110
5.10.4	IAP Groups.....	112
5.10.5	self.add.....	114
5.10.6	self.create.....	115
5.10.7	self.delete.....	116
5.10.8	self.provision.....	117
5.10.9	self.remove.....	118
5.11	Handle Allocation	118
5.11.1	Definition	118
5.11.2	Handle Allocation Service	119
5.11.3	Handle Request.....	120
5.11.4	Handle Response.....	120
5.11.5	Example.....	121
5.12	Interface Blocks.....	121
5.12.1	Definition	121
5.12.2	Interface Topics.....	122
5.12.3	Block Object Properties.....	122
5.12.4	Datapoint Objects	123
5.12.5	Datapoint Presets	130
5.12.6	Datapoint Localization.....	138
5.12.7	Monitor Preference Object	143
5.12.8	Working with Datapoint Values	147
5.12.9	Working with Unions.....	151
5.13	License Services.....	152
5.13.1	License Service Overview.....	152
5.13.2	Actions	152
5.13.3	Capacity	158
5.13.4	Cloud.....	160
5.13.5	Configuration.....	162
5.13.6	License Management	163
5.13.7	Status.....	165
5.14	Load Do Action	167
5.14.1	Definition	167
5.14.2	Outer Image File	167
5.14.3	Load Procedure	168
5.14.4	The Load Object.....	169
5.14.5	Inner Image File Example	169
5.14.6	Load Action and Manifest Properties	170
5.15	On-Demand Monitoring.....	176
5.15.1	Definition	176
5.15.2	On-Demand Datapoint Monitoring Request.....	177
5.15.3	On-Demand Datapoint Monitoring Response.....	178
5.15.4	On-Demand Monitoring Service Collaboration.....	178
5.15.5	Item Poll Request.....	179
5.15.6	Item Poll Response.....	179
5.16	Query	180
5.16.1	Definition	180
5.16.2	Query Request Properties	180
5.16.3	Query Response.....	182
5.16.4	Filtering.....	183
5.17	Schedule Services.....	184
5.17.1	Schedule Services Overview.....	184
5.17.2	Schedule Algorithm.....	185

5.17.3	Schedule Object	186
5.17.4	Calendar Object	189
5.17.5	Calendar Status Object	190
5.17.6	Dates in Schedules and Calendars	192
5.17.7	Specifying Dates	192
5.17.8	Weekly Schedule	196
5.17.9	Exception Schedule	197
5.18	Segment Configuration	199
5.18.1	Definition	199
5.18.2	Segment Configuration Properties	200
5.18.3	Location Object	205
5.19	Segment Do Actions	206
5.19.1	Definition	206
5.19.2	Action Objects	206
5.20	Segment Status	213
5.20.1	Definition	213
5.20.2	Example	213
5.20.3	Properties	214
6	IAP/REST API	214
6.1	IAP/REST Overview	214
6.2	Access	214
6.2.1	URI Definition	214
6.2.2	Query Parameters	216
6.3	Alarms	216
6.3.1	URI Definition	216
6.3.2	Query Parameters	231
6.4	Authentication (Login/Logout)	231
6.4.1	URI Definition	231
6.4.2	Query Parameters	233
6.4.3	Example	233
6.5	Capabilities	234
6.5.1	URI Definition	234
6.5.2	Query Parameters	237
6.6	Connection	237
6.6.1	URI Definition	237
6.6.2	Parameters	237
6.6.3	Query Parameters	237
6.6.4	Examples	238
6.7	Context	238
6.7.1	Context Overview	238
6.7.2	Context - Contextual Entity Relationship	238
6.7.3	Device Assignment	239
6.7.4	URI Definition	239
6.8	Customers	245
6.8.1	URI Definition	245
6.8.2	Query Parameters	249
6.8.3	Examples	249
6.9	Datapoint Categories	250
6.9.1	URI Definition	250
6.9.2	Queries	250
6.10	Datapoint Default Values	252
6.10.1	URI Definition	252

6.10.2	Queries.....	254
6.10.3	Example.....	256
6.11	Datapoint Favorites	257
6.11.1	URI Definition.....	257
6.11.2	Query Parameters.....	259
6.12	Datapoint Logs	259
6.12.1	Overview	259
6.12.2	URI Definition.....	260
6.12.3	Queries.....	264
6.12.4	Logging Configuration	265
6.12.5	Datapoint Log Value.....	265
6.13	Datapoint Monitor	267
6.13.1	Overview	267
6.13.2	URI Definition.....	268
6.13.3	Queries.....	271
6.14	Datapoint Overrides	273
6.14.1	Overview	273
6.14.2	URI Definition.....	273
6.14.3	Queries.....	275
6.15	Datapoint Override State	277
6.15.1	URI Definition.....	277
6.15.2	Queries.....	278
6.16	Datapoint Override Value	279
6.16.1	Overview	279
6.16.2	IAP/MQ Mapping.....	280
6.16.3	URI Definition.....	280
6.16.4	Queries.....	281
6.16.5	Example.....	282
6.17	Datapoint Properties.....	283
6.17.1	URI Definition.....	283
6.17.2	Queries.....	284
6.17.3	Examples.....	285
6.18	Datapoint Tags.....	287
6.18.1	Overview	287
6.18.2	URI Definition.....	287
6.18.3	Queries.....	288
6.19	Datapoint Type	290
6.19.1	Overview	290
6.19.2	IAP/MQ Mapping.....	290
6.19.3	URI Definition.....	290
6.19.4	Queries.....	293
6.20	Datapoint Value.....	295
6.20.1	URI Definition.....	295
6.20.2	Queries.....	296
6.20.3	Examples.....	298
6.21	Device Capabilities	300
6.21.1	URI Definition.....	300
6.21.2	Query Parameters.....	304
6.22	Device Types.....	305
6.22.1	URI Definition.....	305
6.23	Devices.....	318
6.23.1	URI Definition.....	318
6.23.2	Query Parameters.....	347

6.23.3 Example.....	349
6.24 Diagnostics.....	351
6.24.1 URI Definition	351
6.24.2 Query Parameters	352
6.25 DLA Files	352
6.25.1 URI Definition	352
6.25.2 Queries	355
6.26 Drivers.....	358
6.26.1 URI Definition	358
6.27 Floorplans	362
6.27.1 URI Definition	362
6.27.2 Query Parameters	367
6.28 Geozone.....	367
6.28.1 URI Definition	367
6.28.2 Query Parameters	376
6.29 Groups.....	376
6.29.1 URI Definition	376
6.29.2 Group Queries.....	385
6.30 Log Destination.....	388
6.30.1 URI Definition	388
6.30.2 Query Parameters	388
6.31 Log Events.....	389
6.31.1 Log Events Overview	389
6.31.2 URI Definition	389
6.31.3 Query Parameters	392
6.31.4 Event Log Path Parameters	392
6.31.5 Examples.....	394
6.32 Log State.....	396
6.32.1 URI Definition	396
6.32.2 Query Parameters	397
6.33 Log Status.....	397
6.33.1 URI Definition	397
6.33.2 Query Parameters	398
6.33.3 Example.....	398
6.34 On-Demand	398
6.34.1 URI Definition	398
6.35 Permissions	399
6.35.1 URI Definition	399
6.35.2 Query Parameters	400
6.35.3 Example Response.....	400
6.36 Presets	400
6.36.1 URI Definition	400
6.36.2 Query Parameters	402
6.37 Protocols.....	402
6.37.1 URI Definition	402
6.38 Reports	406
6.38.1 URI Definition	406
6.38.2 Query Parameters	416
6.39 Roles.....	417
6.39.1 URI Definition	417
6.39.2 Query Parameters	418
6.40 Schedule Service	418
6.40.1 URI Definition	418

6.40.2	Schedule Service Query and Path Parameters	424
6.40.3	Schedule Path Parameters.....	424
6.40.4	Schedule Service Examples	425
6.41	SMTP Configuration	429
6.41.1	URI Definition.....	429
6.41.2	Query Parameters.....	430
6.42	Transformation Service.....	431
6.42.1	URI Definition.....	431
6.43	Trend Log.....	432
6.43.1	URI Definition.....	432
6.44	Users	435
6.44.1	URI Definition.....	435
6.44.2	Query Parameters.....	440
6.44.3	Examples	440
6.45	Version.....	441
6.45.1	URI Definition.....	441
6.45.2	Query Parameters.....	442
6.45.3	Example.....	442
6.46	Zones	443
6.46.1	URI Definition.....	443
6.46.2	Query Parameters.....	448
6.46.3	Zone Path Parameters.....	448
7	IAP/WS API.....	449
7.1	IAP/WS Overview.....	449
7.2	General Purpose WebSocket.....	449
7.3	Data Log WebSocket.....	449
7.4	Examples	450
7.4.1	Example 1	450
7.4.2	Example 2	451
	Bibliography	452

European foreword

This document (prEN 14908-10:2023) has been prepared by Technical Committee CEN/TC 247 "Building Automation, Controls and Building Management", the secretariat of which is held by SNV.

This document is currently submitted to the CEN Enquiry.

This document is part of the EN 14908 series, which consists of the following parts:

- EN 14908-1, *Open Data Communication in Building Automation, Controls and Building Management — Control Network Protocol — Part 1: Protocol Stack*
- EN 14908-2, *Open Data Communication in Building Automation, Controls and Building Management — Control Network Protocol — Part 2: Twisted Pair Communication*
- EN 14908-3, *Open Data Communication in Building Automation, Controls and Building Management — Control Network Protocol — Part 3: Power Line Channel Specification*
- EN 14908-4, *Open Data Communication in Building Automation, Controls and Building Management — Control Network Protocol — Part 4: IP Communication*
- EN 14908-5, *Open Data Communication in Building Automation, Controls and Building Management Implementation Guideline — Control Network Protocol — Part 5: Implementation*
- EN 14908-6, *Open Data Communication in Building Automation, Controls and Building Management — Control Network Protocol — Part 6: Application Elements*
- EN 14908-7, *Open communication in building automation, controls and building management — Control Network Protocol — Part 7: Communication via internet protocols*
- EN 14908-8, *Open Data Communication in Building Automation, Controls and Building Management — Control Network Protocol — Part 8: Communication using Broadband over Power Line Networks – with internet protocols*
- EN 14908-9, *Open Data Communication in Building Automation, Controls and Building Management — Control Network Protocol — Part 9: Wireless Communication in ISM bands*

Introduction

This document specifies the Internet of Thing IoT Access Protocol (IAP) as a standard web services application protocol based on MQTT (Message Queuing Telemetry Transport), and REST (Representational State Transfer) and WebSockets over HTTP or HTTPS transport protocols.

MQTT, REST, and WebSockets are widely used today for the IoT, but they are low-level protocols with no standardization in the application data and service requests sent using the protocols. IAP delivers that standardization for control and automation networks.

IAP provides secure and open access to the most complex building automation and control systems, enabling integration of existing controls and automation systems with emerging analytics and AI applications that can transform operations from a closed single-protocol loop to an intelligent multi-protocol system.

EN 14908-10 IoT Access Protocol (IAP) web services are an open and extensible standard designed for residential, commercial, and industrial control and automation applications using the EN 14908-1 control network protocol and related protocols to provision and manage IoT devices, to access and update data from the devices, and to aggregate data from diverse devices and protocols for delivery to external applications and services.

IAP web services are typically implemented on a central gateway or edge server that communicates with multiple sensor, actuator, and controller edge devices using one or more drivers such as EN 14908-1, and also interfaces with one or more enterprise and cloud services or applications. The following diagram illustrates a gateway or edge server that uses IAP to communicate with edge devices and cloud services.

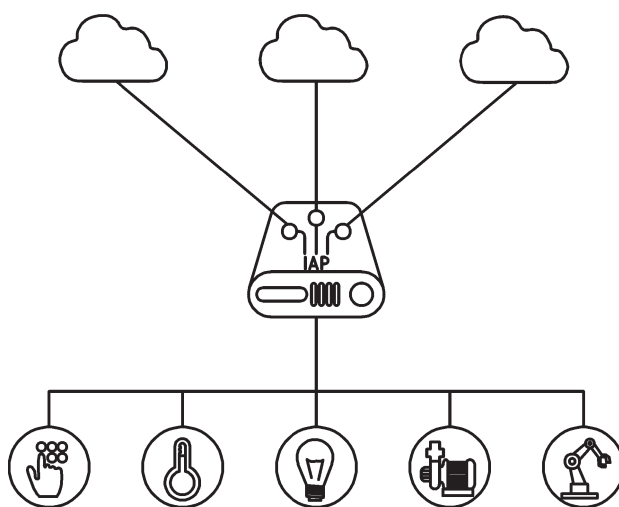


Figure 1

This document is for software developers and web-page authors. It documents three APIs: *IAP/MQ*, which uses MQTT (Message Queuing Telemetry Transport) as the transport protocol, *IAP/REST* (*RE*presentational *S*tate *T*ransfer), which uses REST over HTTP or HTTPS, and *IAP/WS* (WebSockets), which uses WebSockets over HTTP or HTTPS. These APIs are typically implemented on a gateway or edge server that interfaces with edge devices such as energy meters, motion sensors, temperature sensors, HVAC controllers, water heaters/boilers, lighting controllers, refrigeration controllers, variable-frequency drives, motion controllers, and switches. The gateway or edge server typically communicates with the edge devices using a variety of building controls communication protocols including EN 14908-1 control network protocol, BACnet (ISO 16484-5), Modbus, EnOcean, and LoRaWAN.

This document consists of the following four main sections:

- IAP API Fundamentals;
- IAP/MQ API;
- IAP/REST API;
- IAP/WS API.

1 Scope

This document specifies an open and extensible standard for residential, commercial, and industrial control and automation applications using the EN 14908-1 control network protocol and related protocols (EN 14908-2 to EN 14908-9) to provision and manage IoT devices, to access and update data from the devices, and to aggregate data from diverse devices and protocols for delivery to external applications and services.

The web services as specified in this document are typically implemented on a central gateway or edge server that communicates with multiple sensor, actuator, and controller edge devices using one or more drivers such as EN 14908-1, and typically also interfaces with one or more enterprise and cloud services or applications.

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.

EN 14908-1, *Open Data Communication in Building Automation, Controls and Building Management — Building Network Protocol — Part 1: Protocol Stack*

EN 14908-5, *Open Data Communication in Building Automation, Controls and Building Management Implementation Guideline — Control Network Protocol — Part 5: Implementation*

EN 14908-6, *Open Data Communication in Building Automation, Controls and Building Management Implementation Guideline — Control Network Protocol — Part 6: Application elements*

3 Terms and definitions

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

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

— ISO Online browsing platform: available at <https://www.iso.org/obp/>

— IEC Electropedia: available at <https://www.electropedia.org/>

3.1

Message Queuing Telemetry Transport

MQTT

lightweight publish-subscribe, machine to machine network protocol for Message queue/Message queuing service

3.2

Representational State Transfer

REST

software architectural style that describes a uniform interface between physically separate components, often across the Internet in a client-server architecture

3.3

Internet Access Protocol

IAP

open and extensible protocol designed for the Internet of Things to provision and manage devices, and to access and update data from the devices

4 IAP API FUNDAMENTALS

4.1 IAP API Overview

The IoT Access Protocol (IAP) is an open and extensible protocol designed for the IoT to provision and manage devices based on the EN 14908-x protocol suite, to access and update data from the devices, and to aggregate data from diverse devices for delivery to external applications and services such as data analytics and machine learning services. IAP can also be used to support other non- EN 14908-x devices through the methodologies defined in this document.

IAP provides an information model and a data model for the IoT that enables the construction of digital twins of the physical IoT devices in a system. IAP enables these digital twins to be organized into contexts. This organization simplifies integration of the IoT devices to interact with each other and with external services for system monitoring, supervisory control, data analytics, and machine learning.

System monitoring, control, data analytics, machine learning, and AI systems require knowledge of the meaning of the data received from or sent to IoT devices, as well as providing a means to specify the relationships of the data on multiple devices. For example, an IoT device may have a temperature sensor. If the sensor data is exposed as a number with no additional semantics by an edge server, it is difficult for a monitoring and control application to use the temperature value without additional information from the device manufacturer, and it is difficult for an analytics or machine learning application to analyze the value over time without the additional information. A first level of meaning is provided by IAP data types which specify data encoding, size, scaling, units, and usage for a datapoint. Hundreds of standard IAP data types are defined by EN 14908-6, and IAP defines a standard protocol for defining new data types when an appropriate standard data type is not available. For example, the value received from a temperature sensor may be specified using a standard temperature IAP data type that specifies the value is a temperature value expressed in Celsius units.

While an open standard for defining and specifying data types is required, it is not sufficient for supporting an open model for system monitoring, control, analytics, and machine learning applications. For example, a space comfort controller in a building may require multiple temperature sensors to monitor space temperature, supply air temperature, and outdoor air temperature. The data type for these sensors is not sufficient to monitor, control, or analyze the operation of the space comfort controller since all three values may be reported as Celsius temperature values. An open method to differentiate the three types of temperature values is required. IAP provides an open method to create a digital twin of a logical subsystem in an IoT device. The subsystem digital twins are exposed as IAP blocks, where a block is a collection of datapoints with specified meaning. IAP blocks are defined by IAP profiles. More than one hundred standard IAP profiles are defined by EN 14908-6. These profiles were developed by industry-specific working groups to ensure the profiles met the requirements of the specific applications enabled by them.

For example, the space comfort controller can be modelled as a block defined by the IAP standard space comfort controller profile. The IAP space comfort controller profile defines standard inputs for space temperature, source air temperature, and outdoor air temperature. Monitoring, control, and analytics applications can differentiate the three temperature values based on the standard model of the digital twin specified by the IAP profile.

To simplify system integration, IAP extends the digital twin model to the full device. A device is a physical entity with a communications interface. There are two types of devices--edge devices and edge servers. Edge devices typically have one or more sensors, actuators, and controllers. The digital twin for a device contains one or more blocks, each with one or more datapoints. Edge servers provide a unified interface to one or more edge devices. IAP provides device-level operations such as provisioning, testing, and replacement.

To reduce the time required to integrate a system, IAP adds context information for each device, and includes services to leverage the context information when integrating a system. A context can be a

region, campus, building, floor, room, or other area that contains a device. Contexts are organized hierarchically so that a campus contains buildings and areas, a building contains floors, and a floor contains rooms and areas. IAP operations can be specified by templates that can be associated with contexts. For example, IAP supports connection templates that specify how specific datapoints in a context can be connected to other datapoints in the context. This enables the integrator to connect all the temperature sensors in an area to a controller in the area that calculates the average temperature reported by the sensors. The integrator can apply hundreds of area contexts to the connection template to create hundreds of independent connections for each of the areas as defined by the connection template, all with one simple operation.

This standard defines three APIs: IAP/MQ, which uses MQTT (Message Queuing Telemetry Transport) as the transport protocol, IAP/REST (REpresentational State Transfer), which uses HTTP and HTTPS, and IAP/WS API, which uses WebSockets. These APIs are defined in the following sections:

- IAP/MQ Fundamentals;
- IAP/MQ Topic Syntax;
- IAP/REST Fundamentals;
- IAP/REST Syntax;
- Queries and Parameters;
- IAP/WS Fundamentals;
- Data Log Websocket Queries and Parameters.

4.2 IAP/MQ Fundamentals

4.2.1 IAP/MQ Overview

IAP/MQ is the IAP API based on MQTT version 3.1.1. MQTT is an ISO standard (ISO/IEC PRF 20922) publish-subscribe-based messaging protocol. It works on top of the TCP/IP protocol. It is a lightweight protocol that is suitable for implementing applications, services, and drivers on a wide range of devices and servers including small code footprint edge devices, edge servers, and cloud hosts. IAP/MQ is used to connect services and drivers within an edge server and is used to connect edge servers to each other and cloud applications and services

4.2.2 MQTT in IAP

The IoT Access Protocol (IAP) uses an MQTT message broker such as Mosquitto or ActiveMQ, known as the IAP message broker. An IAP system may include central management software (CMS) that provides a user interface for managing the system operation. It includes an IAP message broker, known as the CMS message broker or the CMS broker. IAP edge servers typically include a local message broker, sometimes referred to as a segment message broker.

IAP does not require a specific broker, specific MQTT client library, or implementation. The chosen broker and clients shall support the appropriate versions of MQTT as indicated in the section Supported MQTT Version, below.

4.2.3 Supported MQTT Version

All MQTT components of an IAP system must support MQTT protocol version 3.1.1. MQTT version 3.1 or earlier is not supported.

4.2.4 IAP/MQ Topics

All IAP topics use a specific format as described in **4.4 IAP/MQ Topic Syntax**.

4.2.5 Persistent Clients vs Transient Clients

An IAP client can implement a persistent or a transient MQTT client. Most IAP clients are persistent clients.

For a persistent client, the broker maintains events while the client is unavailable, whereas events are lost to a transient client while the transient client is disconnected.

For example, a client subscribes to a certain set of topics in the IAP/MQ event channel. Then this client disconnects, and later re-connections. Publications made to the subscribed topics during the client's downtime will be delivered to the client if it is a persistent client. A transient client will not see publications that occurred during its downtime.

But a side effect of a persistent client that never reconnects, is that the client can accumulate many messages that the broker keeps on that client's behalf. This issue is addressed in **4.2.5 Client Expiration**.

MQTT clients implement persistence using a constant and unique client ID, and the MQTT property `cleanSession=False` when connecting to the IAP message broker. Persistent clients must specify a unique MQTT client ID and must re-use the same client ID in every session. An MQTT client identifier is between 1 and 23 characters long, UTF-8 encoded, and can consist of letters a-z, A-Z, digits 0-9. Some brokers also allow for additional characters; but that's not part of IAP/MQ. A client ID can be constructed from a combination of the MACID and a sequence number, or completely at random, and is often allocated by agreement among components.

Temporary tools, such as mobile servicing and diagnostic tools, may connect as transient clients, using a unique client ID and the MQTT property `cleanSession=True` when connecting. This includes tools that do not expect to re-connect to the same system regularly or do not need to catch-up with the history of events and changes that occurred between connections.

Transient clients shall specify a unique MQTT client ID but can allocate a new client ID for every session, if necessary. A transient client ID begins with a unique identifier, which has the same format as a persistent client ID.

4.2.6 Client Expiration

A typical IAP message broker supports persistent client expiration at an interval of 2 hours. When a persistent client is disconnected for a duration exceeding this interval, the persistent client's session is automatically cleared to free resources in the broker.

4.2.7 Message Payload

The payload for IAP/MQ messages shall be JSON encoded. However, for readability, many of the examples in this document are shown in Javascript format, which eliminates the use double-quotes around keys, as shown in the following example.

Javascript	JSON
<pre>{ value: 1234, name: "John Denver" }</pre>	<pre>{ "value": 1234, "name": "John Denver" }</pre>

4.2.8 CMS Connectivity

Connectivity between the CMS broker and individual segment brokers is not always present. Some systems may be designed to connect on a schedule and for limited duration only, other systems may simply use unreliable connections. Tools cannot assume that the edge server is present at the time the tool makes a change. Tools can detect the presence or absence of connectivity and indicate the state in the user interface. However, a tool shall not expect to receive immediate confirmation for a change through the feedback channel.

IAP tools shall publish changes to the request channel as necessary and shall respond to updates received through the feedback channel as these updates become available. Updates received from the feedback channel shall report the current system state at the time of the update, which may differ from the state that was requested with the request channel. This difference may be caused by an error processing the request or by a state change initiated by another client to the same system.

For example, a user interface tool may relay a human operator's requests to turn a light on at 50 % brightness. When the tool makes that request, it may reflect the new state of on at 50 % as a tentative state, as yet unconfirmed. When the corresponding feedback is received, the tool can then update its user interface to show the confirmed state, which may not be on at 50 %.

4.2.9 An IAP Site

An IAP-based system consists of one or more sites. A site is typically a physical location, often providing shared Internet access to components within the site. Each site in an IAP-based system includes one or more segments, with one edge server controlling each segment.

4.3 Partial Object Assignment

4.3.1 Definition

IAP/MQ supports partial object assignment where appropriate. For example, a write request may request to a change affecting one datapoint within a device's interface block, leaving other properties and datapoints within the same block unchanged.

Some objects cannot support partial write operations. For example, a datapoint value is considered an atomic value as described in **5.11.8.8 Values are Atomic** and can only be modified whole. For information on datapoint values, device interface blocks, or device objects—see **5.11 Interface Blocks**.

Where partial assignments are supported, empty assignments are also supported. An empty assignment is essentially an assignment without a value change, triggering the renewed publication of the corresponding feedback item. The edge server decides whether such an empty assignment also affects the edge device.

Field and object assignments must be compatible, according to the following rules:

- Some properties may be read only. Tools may write these properties if the value does not change.
- Some properties may be write only. Such a property is used explicitly in an assignment request within the request channel but is not explicitly reported within the feedback channel.
- Some objects and operations require a minimum set of properties. For example, the device `do` topic accepts an `action` object. This object always requires an `action` property, which specifies the desired action (provision, delete, etc.), and the selected action typically requires further mandatory arguments.
- These operations fail unless the minimum required details are provided. Failure is reported with an error diagnostic.

- Some clients may not be aware of all properties received with an object. Clients must not act on object types and topics they don't recognize, but a user interface client could list this data in some generic representation for expert users.
- Privileged clients can add properties to standard objects. Clients must include this property when publishing and ignore it unless it is known and understood.
- User interface components, such as the CMS, do not generally have the privilege to add properties.
- The feedback channel always reports the complete object, even though objects support partial assignment in the request channel.

4.3.2 Edge Servers and Partial Object Assignment

Wherever partial object assignment is supported, edge servers must also support request topics that reach into an object. For example, the `glp/0/bw/rq/dev/lon/1/if/light/0` topic addresses a particular interface block named "light" within a device. This request topic accepts assignment of a complete or a partial light interface block object.

The following shorthand notation illustrates the assignment of a partial object to this topic, affecting only the `desc` property within that block:

```
glp/0/bw/rq/dev/lon/1/if/light/0 ::= { desc: "an example description" }
```

The following example yields the same result, but this time including the partial object in the topic:

```
glp/0/bw/rq/dev/lon/1/if/light/0/desc ::= "an example description"
```

Topics can address items at any depth within the addressed object. For example, the `light` interface block might implement an `sp` datapoint. The Datapoint Object (see **5.11.4 Datapoint Objects**) supports a `monitor` object to control datapoint monitoring, which in turn supports a `rate` property (see **5.11.7 Monitor Preference Object**) to specify the frequency at which a new actual value should be obtained.

The following alternative forms all produce the same result (setting the rate to 17) using partial object assignments:

```
glp/0/bw/rq/dev/lon/1/if/light/0 ::= { sp: { monitor: { rate: 17 } } }
glp/0/bw/rq/dev/lon/1/if/light/0/sp ::= { monitor: { rate: 17 } }
glp/0/bw/rq/dev/lon/1/if/light/0/sp/monitor ::= { rate: 17 }
glp/0/bw/rq/dev/lon/1/if/light/0/sp/monitor/rate ::= 17
```

4.3.3 Timestamps in IAP

The standard timestamp format in IAP is the MRU timestamp (described below). In addition to this timestamp, data events can also use the `ts` format. The `ts` timestamp is a simplified extended ISO 8601 timestamp in the `YYYY-MM-DDTHH:mm:ss.sssZ` format, always in UTC.

The MRU timestamp (describe below) uses the same format as the `ts` format:

`YYYY-MM-DD HH:mm:ss.sss ZZZ`

Where:

- *YYYY* is the four-digit year,
- *MM* is the two-digit month,
- *DD* is the two-digit day of the month,
- *HH* is the hour of the day, always using the 24-hour clock,
- *mm* the number of minutes in the hour,
- *SS* the number of seconds in the minute,
- *SSS* the number of milliseconds within the second,
- *ZZZ* is the name of the time zone for wall time or UTC.

For information on configuring the time zone, see the `time` object described in **5.17 Segment Configuration**.

All numbers in a timestamp are padded with 0 to the left to meet the specified number of digits, if necessary. Number groups are separated with a single hyphen, colon, period, or a single space character.

EXAMPLES

```
2018-11-30 16:33:05.257 BST
2019-01-09 15:02:18.321 UTC
```

4.3.4 Most Recently Used (MRU) Timestamp

All messages published in the feedback channel must include a most recently used (`mru`) timestamp, using the standard IAP timestamp format. Messages posted in other channels may include an MRU timestamp. All MRU timestamps are UTC timestamps.

Most recipients of messages ignore the MRU, but recipients may use the MRU, for example, to prune stale objects which may have resulted from the deletion of items. MRU timestamps are always ignored when assigned through the request channel.

Deletion of resources generally leaves one item behind, a status object which informs clients that reconnect in the future that the resource in question has been deleted. These items are typically pruned on a low frequency.

4.4 IAP/MQ Topic Syntax

4.4.1 Syntax elements

The syntax for topics is as follows:

```
glp/{ IAP_Ver}/{SID}/{Channel}/{Resource_Type} [{Resource_Detail}] [{Property_Path}]
```

EXAMPLE

glp/0/s1/fb/dev/lon/2/if/light/1

Table 1 describes the IAP/MQ syntax levels.

Table 1 — IAP/MQ Topic Syntax

Level	Description
glp	IAP topics always begin with glp, where glp is a constant keyword. A topic cannot begin with a slash.
IAP_Ver	The topic must specify the IAP version. This standard describes version 0 (zero). All IAP implementations shall support version 0 of the protocol. The number provided here indicates the highest protocol version supported. IAP supports all versions up to and including the maximum protocol version indicated with glp/0/{SID}/about/version.
SID	<p>The SID is the segment identifier, which is the logical address of the device that controls a segment. In IAP, that device is either an edge server, or, an IAP-native edge device that is the only device on that segment. Such an edge device can implement other object types, such as alarms or connections for itself, and uses the standard topic layout.</p> <p>An SID consists of the letters A-Z, a-z, digits 0-9, and special characters period (.), hyphen (-), and underscore (_). Special characters cannot be used as the first character of a SID. A valid SID matches the following regular expression:</p> <pre>[A-Za-z0-9] [A-Za-z0-9.-_]*</pre> <p>The SID is assigned to the edge server when the segment is provisioned. A regular SID has the format shown above. A different address is used in place of the SID in certain specific circumstances, such as before the segment is provisioned, or to indicate a local address for the edge server. For information about SIDs before an edge server is provisioned, see 5.7.2 Segment Discovery.</p> <p>A local address for the edge server can be used for communication between IAP components within a segment. Topics using "." for the SID are called local topics. Local topics are not relayed to the CMS or other segments through bridges. Topics using local addressing never traverse connections between CMS and a remote edge server.</p> <p>A period identifier used outside the segment represents a segment-independent address. And like local topics, segment-independent topics are not relayed across bridge connections. A period identifier is used for certain services, such as the logging service described in 6.12 Datapoint Logs.</p> <pre>glp/0/./=logger/event</pre> <p>The = sign indicates that the topic is addressing a service, instead of an object. Changing a segment identifier requires that the segment be deprovisioned first, then re-provisioned with the new segment identifier.</p> <p>Reserved SIDs:</p> <ul style="list-style-type: none"> Beginning an SID with one of the following characters is reserved for use within the edge server: %, &, \$ No SID can begin with # or + to avoid confusion with MQTT topic

Level	Description
	wildcards. — No SID can contain a slash (/).
<i>Channel</i>	<p>IAP groups topics and message handling attributes into directional communication channels. Additional channels may exist within an edge server, but tools must not react to channel types that they are unaware of. When necessary, tools must process any unknown channel types verbatim. IAP includes the following channels, with the following identifiers:</p> <p><i>rq</i></p> <p>Request Channel. Tools use the request channel to publish messages requesting value or state changes. Messages in the request channel flow from the tool or CMS to the edge server, and within the segment the messages flow to the edge processor or to another service. The request flows to the component responsible for acting on it.</p> <p>All publishing to the request channel use MQTT properties QoS 2, retain=False.</p> <p><i>fb</i></p> <p>Feedback Channel. The feedback channel provides the system state from the edge processor or another component within the edge server to all clients.</p> <p>All publishing to the feedback channel uses MQTT properties QoS=1, retain=True.</p> <p>The <i>fb</i> channel is retained so that new clients receive the current system state when they subscribe to the feedback channel. A client can subscribe to the entire feedback channel, or only to selected topics and topic groups within the feedback channel.</p> <p>All feedback channel objects include a timestamp of the most recent update for that object. This is provided in the <i>mr_u</i> property.</p> <p><i>ev</i></p> <p>Event Channel. The event channel provides diagnostics, such as errors, warnings, alarms, and similar information to all clients. All publishing to the event channel use MQTT properties QoS=1, retain=False.</p> <p>A new client will not receive events that occurred in the past, and neither will a disconnected non-persistent client. However, a disconnected persistent MQTT client will receive events missed while it was disconnected. A persistent MQTT client is one that connects with the MQTT "cleanSession" attribute set to False.</p> <p><i>bin</i></p> <p>Binary Channel. Topics within the binary channel address a binary blob of data. There are no objects or properties within objects in the binary channel, and binary channel topics do not provide any hierarchy within the channel.</p> <p>An IAP/MQ binary channel topic matches the following filter:</p> <div>glp/0/+ /bin/+ /#</div> <p>An IAP/MQ binary channel topic typically takes the form:</p> <div>glp/0/{SID} /bin/{Handle}</div>

Level	Description
	<p>Additional topic elements are possible, such as: <code>glp/0/{SID}/bin/{Handle}/a/b/</code></p> <p>Unlike the request, feedback, or event channel, a binary channel topic, such as <code>glp/0/{SID}/bin/{Handle}/a/b</code>, is independent and has no relation to <code>glp/0/{SID}/bin/{Handle}/a</code>.</p> <p>Blobs of data within the binary channel are limited in size and are usually considered short-lived. A use-case is that of embedded transfer of small binary images through IAP/MQ for use with the load action described in 5.13 Load Do Action.</p> <p>IAP/MQ infrastructure components support at least three simultaneous publications up to 128 KB, each within the binary channel. Publications exceeding the size limit, or exceeding the maximum number of simultaneous publications, can be refused with standard MQTT message codes, using <code>no PUBACK</code> with MQTT 3.1.1, or a negative acknowledgment with MQTT 5.</p> <p>Simultaneous publications are those that are published with <code>retain=true</code> and are not yet deleted, plus those published with <code>retain=false</code> and not yet delivered to all subscribers.</p> <p>Items are published into the binary channel using <code>QoS=1</code> or <code>2</code>, <code>retain=true</code>. The consumer of that data, such as the component executing a load action, is expected to remove that data by publishing <code>null</code> with <code>retain=true</code> to the same topic.</p>

Level	Description																								
Resource_Type	IAP supports a variety of resource types such as devices, groups, schedules, etc. The supported resource types include those in the following list.																								
	<table><tr><th>Resource Type</th><th>Resource</th></tr><tr><td>about</td><td>Segment Information</td></tr><tr><td>alarm</td><td>Alarms</td></tr><tr><td>app</td><td>Apps</td></tr><tr><td>cfg</td><td>Segment Configuration</td></tr><tr><td>con</td><td>Connections</td></tr><tr><td>dev</td><td>Devices (see 4.4.2 Device Syntax below)</td></tr><tr><td>grp</td><td>Groups</td></tr><tr><td>res</td><td>Resource Data Types</td></tr><tr><td>sev</td><td>Schedule Services</td></tr><tr><td>sts</td><td>Segment state</td></tr><tr><td>=system</td><td>Handle request The = sign is used to indicate that the topic is addressing service, instead of an object.</td></tr></table>	Resource Type	Resource	about	Segment Information	alarm	Alarms	app	Apps	cfg	Segment Configuration	con	Connections	dev	Devices (see 4.4.2 Device Syntax below)	grp	Groups	res	Resource Data Types	sev	Schedule Services	sts	Segment state	=system	Handle request The = sign is used to indicate that the topic is addressing service, instead of an object.
	Resource Type	Resource																							
	about	Segment Information																							
	alarm	Alarms																							
	app	Apps																							
	cfg	Segment Configuration																							
	con	Connections																							
	dev	Devices (see 4.4.2 Device Syntax below)																							
	grp	Groups																							
	res	Resource Data Types																							
	sev	Schedule Services																							
	sts	Segment state																							
	=system	Handle request The = sign is used to indicate that the topic is addressing service, instead of an object.																							
Resource_Detail	The general syntax for the <i>Resource Detail</i> is: <i>[{Driver_ID}/] {Handle}/{Object_Path}</i> Depending on the resource type, <i>Resource Detail</i> may include a <i>Driver_ID</i> (such as for <i>dev</i> resources), and the <i>Object_Path</i> element can be multiple levels deep, depending on the object being specified. Possible topic syntax for a <i>dev</i> Resource and <i>grp</i> Resource is shown below: For a <i>dev</i> resource: <i>glp/0/{SID}/{Channel}/dev/{Driver_ID}/ {Handle}/{Object_Path}</i> For a <i>grp</i> resource: <i>glp/0/{SID}/{Channel}/grp/{Handle}/{Object_Path}</i> See 4.4.2 Device Syntax below for more syntax information.																								
	Property_Path	Many resources allow topics to address a property within the selected object. This includes nested properties, which are separated with a single forward slash. Consider the following example: <i>glp/0/s1/rq/dev/lon/a/if/LightCntrl/0/nviLampValue/ value</i> Where — <i>glp/0/s1</i> is the segment address; — <i>.../rq</i> addresses the request channel within the segment;																							

Level	Description
	<ul style="list-style-type: none"> — <code>.../dev/lon/a</code> addresses a device object (<code>.../Resource_Type/Driver_ID/Handle</code>); — <code>.../if/LightCntl/0</code> addresses an interface block within the device (<code>.../Object_Path</code>); <code>.../nviLampValue/value</code> is a property path within that interface block object.

4.4.2 Device Syntax

`.../dev / {Driver_ID} / {Handle} / {Object_Path} [ / {Property_Path} ]`

Following the `dev` resource element, device topics can include the elements listed in Table 2:

Table 2 — Device Syntax

Topic Element	Description														
<i>Driver_ID</i>	<p>Topics separate device objects according to their driver. When required, the driver identifier follows the resource type identifier.</p> <p>Edge servers can support multiple drivers. Each supported protocol shall be identified with a unique driver ID. The following driver IDs are specified by this standard – any additional protocols shall use different driver IDs.</p> <table> <tr> <th>Driver ID</th><th>Description</th></tr> <tr> <td>bacnet</td><td>BACnet (ISO 16484-5)</td></tr> <tr> <td>enocan</td><td>EnOcean (ISO/IEC 14543-3-10)</td></tr> <tr> <td>iap</td><td>IAP/MQ (this standard)</td></tr> <tr> <td>lon</td><td>CNP and CNP/IP (EN 14908-1 and EN 14908-7)</td></tr> <tr> <td>lorawan</td><td>LoRaWAN (ITU-T Y. 4480)</td></tr> <tr> <td>modbus</td><td>Modbus</td></tr> </table>	Driver ID	Description	bacnet	BACnet (ISO 16484-5)	enocan	EnOcean (ISO/IEC 14543-3-10)	iap	IAP/MQ (this standard)	lon	CNP and CNP/IP (EN 14908-1 and EN 14908-7)	lorawan	LoRaWAN (ITU-T Y. 4480)	modbus	Modbus
Driver ID	Description														
bacnet	BACnet (ISO 16484-5)														
enocan	EnOcean (ISO/IEC 14543-3-10)														
iap	IAP/MQ (this standard)														
lon	CNP and CNP/IP (EN 14908-1 and EN 14908-7)														
lorawan	LoRaWAN (ITU-T Y. 4480)														
modbus	Modbus														

Topic Element	Description
<i>Handle</i>	<p>Handles are identifiers of addressable items used with IAP topics. The handle is a unique identifier within its scope. For example, every device object has a unique handle within its driver and within its segment.</p> <p>A handle is a case-sensitive string at least one character long and consists of letters A-Z and a-z, digits 0-9, and special characters period (.), hyphen (-), and underscore (_). A period can be used to separate parts of a handle but shall not be used as the first character. The following characters are reserved, and shall not be used: =, :, &</p> <p>The handle is assigned by the tool that creates the primary item. For example, a tool might create a CNP device and assign handle "1" or "2" as shown in the example below.</p> <pre>glp/0/s1/rq/dev/lon/1/ glp/0/s1/rq/dev/lon/2/</pre> <p>Other handles are intrinsic to the principal item. For example, a device might have two <code>light</code> interface blocks, which appear with interface block address <code>light/1</code> and <code>light/2</code>, respectively.</p> <p>Tools allocate handles when creating resources through the request channel. Tools are responsible for allocating a unique handle.</p> <p>For more information, see 5.10 Handle Allocation.</p>
<i>Object_Path</i>	<p><i>Object_Path</i> refers to the object addressed within the device. Depending on the specific object, the <i>Object_Path</i> can be multiple levels deep and can contain additional handles.</p> <p>Each device supports a configuration object with the <code>cfg</code> level in the request and the feedback channel. In the request channel it is used to support modifications to the object configuration. In the feedback channel it always provides the complete and correct configuration object. The following examples show the <code>cfg</code> level in the feedback and request channels.</p> <p>EXAMPLES</p> <pre>glp/0/s1/rq/dev/lon/2/cfg glp/0/s1/fb/dev/lon/2/cfg</pre> <p>Where</p> <ul style="list-style-type: none"><code>lon</code> is the <i>Driver ID</i>,<code>2</code> is the <i>Handle</i>,<code>cfg</code> is the <i>Object Path</i>. <p>Other endpoints within a device are datapoints in the device's interface object (<code>if</code>) defined in 5.11 Interface Blocks, which contains sets of blocks of functionality. For example, one device might support two lights and one occupancy sensor, perhaps addressed with these topics:</p>

Topic Element	Description
	<p>EXAMPLE</p> <pre>glp/0/s1/fb/dev/lon/2/if/light/0 glp/0/s1/fb/dev/lon/2/if/light/1 glp/0/s1/fb/dev/lon/2/if/occ/0</pre> <p>Where</p> <p>lon is the <i>Driver ID</i>, 2 is the <i>Handle</i>, if/light/0, if/light/1, if/occ/0 are the <i>Object_Paths</i>.</p> <p>Most object types support a status object (<i>sts</i>) in the feedback channel, and a <i>do</i> endpoint in the request channel used to invoke methods on the object.</p> <p>Many properties and nested properties within an addressable object can be addressed with the object path, but exceptions exist: a datapoint value, which is within most interface blocks, can only be written in its entirety, and can never be written only in parts. Therefore, a datapoint value can be addressed and written with an object path alone only if the datapoint implements a scalar type. Datapoints implementing complex types must be written with an object path that ends at the <i>value</i> property within the datapoint object.</p> <p>For example: value 1234 can be published to a scalar datapoint with a topic such as <code>glp/0/1/rq/dev/lon/1/if/light/0/sp/value</code>.</p> <p>A datapoint that implements a complex type such as a structure with <i>r</i>, <i>g</i>, <i>b</i> values, must be written in its entirety, for example by writing <code>{r: 0.5, g: 0.7, b: 0.83}</code> to <code>glp/0/1/rq/dev/lon/1/if/light/0/sp/value</code></p>

4.4.3 Wildcards

The following wildcards can be used in the topic syntax for MQTT subscriptions: # and +

Clients can receive messages by creating subscriptions. A subscription may be to an explicit topic, in which case only messages to that topic will be received. Or a subscription may include wildcards. Table 3 describes the two wildcards.

Table 3 — Wildcards

Wildcard	Description
#	Can be used to represent all remaining levels of hierarchy. This means that it must be the final character in a subscription.
+	Can be used to represent a single level of hierarchy in a subscription.

4.5 IAP/REST Fundamentals

4.5.1 REST Overview

REST is an architectural style that allows applications to communicate effectively over HTTP by using HTTP methods (such as GET, PUT, and DELETE) to manage resources. REST is not a specification.

However, it does have a number of constraints that together form the key guidelines for REST. The key constraints leveraged by IAP/REST are:

- *client-server model*, which requires that the responsibilities of the client application are completely separate from those of the server. All the client needs to know to access a service of the server is the resource's URI.
- *stateless*, which requires that the server does not store any client-specific information about the last HTTP request made by the client. Each request made by the client must provide all of the information required to handle the request, either provided as part of the URI, query parameters, body, or headers. The server's response includes the information regarding the state, either included in the headers, response body, or status.
- *uniform interface*, which requires each resource to be identified using a single, unique URI, and all clients access all resources in the same manner.
- *layered system*, which allows you the option to use a layered system architecture, where the client cannot tell whether it is connected directly to the server or to an intermediary, such as a proxy or gateway.

REST components communicate by transferring a representation of a resource in a format as described by this standard, selected dynamically based on the capabilities or desires of the recipient and the nature of the resource.

4.5.2 REST in IAP

IAP/REST is an IoT access protocol based on REST. IAP/REST defines a standard set of resource types that are required for managing, monitoring, and controlling IoT devices. Resource types include devices, datapoints, alarms, logs, and schedules. Many IAP/REST resources represent a physical entity. For example, an IAP/REST device resource represents a physical IoT device. Some resources are contained within other resources. For example, an IAP/REST device resource may contain a number of IAP/REST datapoints, where each datapoint resource represents a datapoint on the physical IoT device defined by the IAP/REST device resource.

4.5.3 Making IAP/REST Calls

A client application shall authenticate with the IAP/REST server prior to making any IAP/REST requests from the server as described in **6.3 Authentication (Login/Logout)**.

4.6 IAP/REST Syntax

The syntax for IAP resources is as follows:

```
/iap/{Resource_Type} [{/ {Resource_Detail}}] [{? {Query_Parameters}}]
```

The syntax notation uses bold to indicate items that must be entered as shown, and terms in italics within parentheses to represent variables that must be replaced with an appropriate value. IAP is case-sensitive, so case must be as shown in the syntax descriptions.

EXAMPLE

```
GET /iap/floorplans/floors/{id}
```

Table 4 summarizes the IAP/REST syntax.

Table 4 — IAP/REST Syntax

Segment	Description																																				
iap	IAP/REST URIs always begin with /iap, where iap is a constant keyword.																																				
<i>Resource_Type</i>	<p>IAP/REST supports a variety of resource types such as devices, alarms, datapoints, etc. The supported resource types includes those in the following list:</p> <table> <tr> <th>Resource Type</th><th>Resource</th></tr> <tr> <td>access</td><td>Geozone</td></tr> <tr> <td>alarms</td><td>Alarms</td></tr> <tr> <td>auth</td><td>Authentication</td></tr> <tr> <td>customers</td><td>Customers</td></tr> <tr> <td>devs</td><td>Devices, Device Capabilities, Datapoints</td></tr> <tr> <td>diagnostics</td><td>Diagnostics</td></tr> <tr> <td>floorplan</td><td>Floorplans</td></tr> <tr> <td>generateKey</td><td>Customers</td></tr> <tr> <td>gz</td><td>Geozone</td></tr> <tr> <td>log</td><td>Log Destination, Log Events</td></tr> <tr> <td>permission</td><td>Permissions</td></tr> <tr> <td>reports</td><td>Reports</td></tr> <tr> <td>role</td><td>Roles</td></tr> <tr> <td>scevent</td><td>Schedule Events</td></tr> <tr> <td>sch</td><td>Schedules</td></tr> <tr> <td>user</td><td>Users</td></tr> <tr> <td>version</td><td>Version</td></tr> </table>	Resource Type	Resource	access	Geozone	alarms	Alarms	auth	Authentication	customers	Customers	devs	Devices, Device Capabilities, Datapoints	diagnostics	Diagnostics	floorplan	Floorplans	generateKey	Customers	gz	Geozone	log	Log Destination, Log Events	permission	Permissions	reports	Reports	role	Roles	scevent	Schedule Events	sch	Schedules	user	Users	version	Version
Resource Type	Resource																																				
access	Geozone																																				
alarms	Alarms																																				
auth	Authentication																																				
customers	Customers																																				
devs	Devices, Device Capabilities, Datapoints																																				
diagnostics	Diagnostics																																				
floorplan	Floorplans																																				
generateKey	Customers																																				
gz	Geozone																																				
log	Log Destination, Log Events																																				
permission	Permissions																																				
reports	Reports																																				
role	Roles																																				
scevent	Schedule Events																																				
sch	Schedules																																				
user	Users																																				
version	Version																																				

Segment	Description
<i>Resource_Detail</i>	<p>Depending on the Resource Type, the <i>Resource Detail</i> may be zero or multiple levels deep, depending on the resource being specified.</p> <p>The following examples show the syntax for a <code>user</code> resource without any <i>Resource Detail</i>, and a <code>device</code> resource with two <i>Resource Detail</i> levels:</p> <p>For a <code>user</code> resource:</p> <pre>GET /iap/user</pre> <p>For a <code>devs</code> resource:</p> <pre>GET /iap/devs/floor/{id}</pre> <p>Resource details can also include item reference fields. For example, the Datapoint Categories resource uses the following URI:</p> <pre>/iap/devs/{id}/if/{block_name}/{block_index}/{datapoint_name}/cat</pre> <p>The item reference fields in this URI are <code>id</code>, <code>block_name</code>, <code>block_index</code>, and <code>datapoint_name</code>. Item reference fields can contain path parameters as described in 4.7 Queries and Parameters.</p>
<i>Query_Parameters</i>	<p>Query parameters can be added to the end of the URI, following a '?' character. Multiple parameters can be specified by separating each parameter with an '&' character. For more information see 4.9 Query Parameters.</p>

4.7 Queries and Parameters

A query specifies a selection rule for a specified property. A query may be specified within a path element of the path component, or within a query component.

- A query specified in a path element is called a path parameter. See **4.8 Path Parameters**.
- A query specified in a query component is called a query parameter. See **4.9 Query Parameters**.

A query in IAP/REST is a key-value pair along with an operator that specifies the relationship between the key and the value. The syntax for a query is `{key}{operator}{value}`. Multiple key-value pairs can be specified, separated by an "&" character as a logical "AND", combining the parameters.

`{key}`, `{operator}`, and `{value}` have the following descriptions:

- *key* – Identifies a property of the IAP/REST resources. When included in a path element, the *key* specifies a property within the IAP/REST object specified by the path element.
- *operator* – Specifies the relation between the *key* and *value*. Table 5 lists the available operators.

Table 5 — Queries and Parameters - Operators

Operator	Definition
==	Exact match
!=	Exact mismatch (for a fully specified string)
=+	Regular expression match, case-insensitive
!+	Regular expression mismatch, case-insensitive
<	Less than (for numbers and timestamps)
<=	Less than or equal (for numbers and timestamps)
=>	Equal or greater than (for numbers and timestamps)
>	Greater than (for numbers and timestamps)

- *value* – Specifies the value of the property to be used to identify selected resources. The value can be:
 - An absolute value for scalar properties.
 - A regular expression for the regular expression match (=+) or regular expression mismatch (!+) operators.

For the regular expression match (=+) and regular expression mismatch (!+) operators, the regular expression syntax is defined as a subset of Perl 5 syntax supported by `java.util.regex` as described in the `java.util.regex` documentation at the following link:

<https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html>.

Regular expressions cannot include characters that are not allowed in URI paths. Any such characters must be percent encoded as described in <https://en.wikipedia.org/wiki/Percent-encoding>. For example, Table 6 lists characters and their encoding that are reserved in [RFC 3986](#), along with additional special characters that also require percent encoding:

Table 6 — Queries and Parameters – Special Characters

Character	Encoding	Special Characters	Encoding
!	%21	newline	%0A or %0D or %0D%0A
#	%23	space	%20
\$	%24	"	%22
&	%26	%	%25
'	%27	-	%2D
(	%28	.	%2E
)	%29	<	%3C
*	%2A	>	%3E
+	%2B	\	%5C
,	%2C	^	%5E
/	%2F	_	%5F
:	%3A	`	%60
;	%3B	{	%7B
=	%3D		%7C
?	%3F	}	%7D
@	%40	~	%7E
[	%5B		
]	%5D		

Examples

This is a simple request for a named datapoint `sp` within a device with ID `myDevice`:

```
GET iap/devs/myDevice/if/Light/0/sp
```

This addresses all datapoints named `sp` within any device whose `type` property matches a regular expression, and whose block type (the profile) is reported as a particular and specific value. All matching blocks are considered regardless of their handle.

```
GET iap/devs/*+type=^90000146164[0-4]85/if/*+type==0000000000000000-0%2FSFPTopenLoopActuator%2F*/sp
```

4.8 Path Parameters

A path parameter can be included in any item reference field for the path component. The item reference fields are the `id`, `block_name`, `block_index`, `datapoint_name`, and `prop_name` values in the following path:

```
/iap/devs/{id}/if/{block_name}/{block_index}/{datapoint_name}/value
[/ {prop_name}]
```

Path parameters may be specified in the following alternate forms:

- A single value that identifies a single object that can be identified by the path element.
- A single asterisk (*), identifying all objects that can be identified by the path element.
- A single asterisk followed by a plus sign (*+) followed by one or more key-value pairs (`{key}{operator}{value}`), each adjacent pair separated by an ampersand (&).

EXAMPLE

Request

```
GET /iap/devs/8/if/Lamp/0/nviValue/value
```

Response

```
[ {
  "deviceId" : 8,
  "blockName" : "Lamp",
  "blockIndex" : "0",
  "datapointName" : "nviValue",
  "value" : {
    "value" : 0,
    "state" : 0
  }
} ]
```

4.9 Query Parameters

Query parameters can be added to the end of the URI, following a '?' character. If more than one query parameter is specified, then each parameter is separated with an '&' character.

Each query parameter has the form `<name>=<value>`, where each value can be specified as a single string value, a simple wildcard by specifying a single asterisk ("*") as the value, or a Perl regular expression, terminated with a semicolon and enclosed in quotes (`"|<Perl regular expression>|;"`).

Table 7 describes the optional query parameters:

Table 7— Query Parameters

Query Parameter	Description
<code>max_age={max_seconds}</code>	For queries that return a datapoint value, <code>max_age</code> specifies the maximum datapoint age for any returned datapoint values. <code>max_age</code> forces a poll request for the datapoint, if the last datapoint value is older than the time specified by <code>max_seconds</code> . The value cannot be specified as a regular expression.
<code>pg={page}</code>	<p>Specifies the collection of resources found on the page specified by <code>page</code>. Null means one (1).</p> <p>If a snapshot number is not provided with the <code>xs</code> key, the CMS creates a new snapshot of the resources identified by the request path and returns the snapshot number to the client.</p> <p>If a valid snapshot number is provided with the <code>xs</code> key, the page is fetched from the specified snapshot.</p> <p>If an invalid snapshot number is provided with the <code>xs</code> key, an error response is returned.</p> <p>If no page number is specified with the <code>pg</code> parameter, all resources identified by the path parameter are returned.</p>
<code>sz={size}</code>	<p>Specifies the number of resources per page to be used for page requests. Must be an integer greater than zero. Null means no limit.</p> <p>If a <code>pg</code> parameter is specified, but the page size is not specified with the <code>xs</code> parameter, the default is 50. For example: 100.</p> <p>Resources may be skipped or duplicated if the requesting application changes the <code>xs</code> value between subsequent requests for different pages.</p>
<code>xs={snapshot_num}</code>	<p>Specifies the snapshot number to use when a page is requested using the <code>pg</code> parameter.</p> <p>When the first of a series of paginated requests is made, the requesting client does not include a snapshot number. When a page is requested using the <code>pg</code> parameter and a snapshot number is not provided with the <code>xs</code> parameter, the requested resources are gathered, sorted in ascending order according to their ID, a new snapshot of the resources is created, and then the snapshot number is returned to the client.</p> <p>When a paginated response is requested with the <code>pg</code> parameter, if a valid snapshot number is provided with the <code>xs</code> parameter, the CMS fetches the page from the specified snapshot.</p> <p>If an invalid snapshot number is provided with the <code>xs</code> parameter, the CMS returns an error response.</p>

4.10 IAP/WS Fundamentals

The IAP/WS API uses WebSockets to support two-way communication between client apps and the IAP server. Unlike REST, which requires the client to poll and then wait for a response, the WebSocket API allows a client to get real-time updates from the IAP server. To use a WebSocket, the client issues a

WebSocket request to a server, the server opens a WebSocket, and then the server sends one or more WebSocket updates to the client.

WebSockets are used to get datapoint updates and to get data logs from an IAP/WS server.

4.11 Data Log WebSocket Queries and Parameters

4.11.1 Data Log WebSocket Overview

Only data log WebSockets use queries and parameters.

That is, the general purpose WebSocket is started by `/iap/ws` and does not use queries or parameters. Once you start a general WebSocket, you can use REST API requests to request on-demand GET requests.

A query specifies a selection rule for a specified object or objects. A query parameter can be specified and added to the end of a URI following a `?` character.

The remainder of this section applies to data log WebSockets.

4.11.2 Key

The *key* identifies a property of the IAP/REST resources. When included in a path element, the *key* specifies a property within the IAP/REST object specified by the path element.

4.11.3 Operator

The *operator* specifies the relation between the *key* and *value*.

4.11.4 Value

The *value* specifies the value of the property to be used to identify selected resources. The value can be:

- An absolute value for scalar properties.
- A regular expression for the regular expression match (`=+`) or regular expression mismatch (`!+`) operators.

For the regular expression match (`=+`) and regular expression mismatch (`!+`) operators, the regular expression syntax is defined as a subset of Perl 5 syntax supported by `java.util.regex` as described in the `java.util.regex` documentation at the following link:

<https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html>.

4.11.5 Regular Expression Encoding

Special characters may need to be encoded before sending the IAP/WS request to the IAP server. Not all instances of a special character need to be encoded. Encoding depends on where the special character is used in the IAP/WS request.

Encoding in IAP/WS is not the same as IAP/REST.

Device, block and datapoint names are not encoded except for spaces. This is different from the REST API which does require special characters in Device, block and datapoint to be encoded.

Regular expressions cannot include characters that are not allowed in URI paths. Any such characters must be percent encoded as described in <https://en.wikipedia.org/wiki/Percent-encoding>. For

example, the following characters from Table 8 that are reserved in RFC 3986 must be encoded as shown below each character:

Table 8 — Regular Expressions - Using Percent Encoding

!	#	\$	&	'	(	)	*	+	,	/	:	;	=	?	@	[	]
%21	%23	%24	%26	%27	%28	%29	%2A	%2B	%2C	%2F	%3A	%3B	%3D	%3F	%40	%5B	%5D

Following are additional special characters that require percent encoding as shown in Table 9:

Table 9 — Special Characters Requiring Percent Encoding

newline	space	"	%	-	.	<	>	\	^	_	`	{		}	~
%0A or %0D or %0D%0A	%20	%22	%25	%2D	%2E	%3C	%3E	%5C	%5E	%5F	%60	%7B	%7C	%7D	%7E

For example, to specify XIF names in a request, instead of instance names, add `*+xifName==` encoded to `%2A%2BxifName==`

&dpName=`*+xifName=={datapoint XIF name}` encoded
to &dpName=`%2A%2BxifName=={datapoint XIF name}`

Example: &dpName=`*+xifName==nvoLuxLevel` encoded
to &dpName=`%2A%2BxifName==nvoLuxLevel`

To specify datapoint instance names without using exact matches add `*+name==` encoded to `%2A%2Bname==`

&dpName=`*+name=={datapoint instance name}` encoded
to &dpName=`%2A%2Bname=={datapoint instance name}`

Example: &dpName=`*+name==nvoLightLevel` encoded
to &dpName=`%2A%2Bname==nvoLightLevel`

The `&value` parameter specifies timespans. The time span request shall be encoded.

Equal sign "=" in name parameters is not encoded but it is in `&value` parameters.

5 IAP/MQ API

5.1 Overview

IAP/MQ is the IAP API based on MQTT. MQTT is an ISO standard (ISO/IEC PRF 20922) publish-subscribe-based messaging protocol. It works on top of the TCP/IP protocol. It is a lightweight protocol that is suitable for implementing applications, services, and drivers on a wide range of devices and servers including small code footprint edge devices, edge servers, and cloud hosts. IAP/MQ is used to connect services and drivers within an edge server and is used to connect edge servers to each other and cloud applications and services.

This section defines the IAP/MQ API topics, which allow access to the commonly used resources.

For information on the IAP/MQ topic syntax, see **4.2.4 IAP/MQ Topics**.

5.2 About

5.2.1 Definition

glp/0/{SID}/fb/about

The About topic informs tools of the IAP server's make and model, capabilities, and limitations. This topic is updated in the feedback channel once when the edge server starts. The topic returns all the available information.

5.2.2 Example

```
{
  "capabilities":["lon-routing-rni", "lon-routing-router", "arbitrary-
alarms", "device-grouping", "scheduler", "history", "pagination-
snapshots"],
  "limitations":{"devices":20,"protocols":["lon","modbus"]},
  "components":[
    {"name":"ltx",
      "desc":"CNP Translator
      "license":"Copyright (c) 2020 XYZ Corporation. All rights
reserved.",
      "manufacturer":" XYZ Corporation",
      "version":"1.14.050"},
    {"name":"lte",
      "desc":"CNP Engine",
      "license":"Copyright (c) 2020 XYZ Corporation. All rights
reserved.",
      "manufacturer":"XYZ Corporation",
      "version":"2.40-386"},
    {"name":"Modbus protocol engine",
      "desc":"Modbus protocol engine ",
      "license":"Copyright (c) 2020 XYZ Corporation. All rights
reserved.",
      "manufacturer":"XYZ Corporation",
      "version":"v1.00.004"}
  ],
  "protocol":{"lon":{"devices":32768},"modbus":{"devices":20}},
  "copyright":"Copyright (c) XYZ Corporation. All rights reserved",
  "discovery":"satm",
  "manufacturer":"XYZ Corporation",
  "model":"XYZ Segment Controller",
  "product":"XYZSC",
  "version":"2.40.414",
  "glpVersion":0,
  "languages":["en"],
  "mru":"2019-02-08 15:12:38.600 UTC"
}
```

5.2.3 Properties

Table 10 lists the About object properties:

Table 10 — About Properties

Field Name	Type	Description
copyright	string	(Optional) Reports the copyright information. EXAMPLE "Copyright (C) 2018 XYZ Corporation"
discovery	string	Reports the supported discovery method. Possible values: none Discovery of devices is not supported. This method is specified if the driver supported by this segment does not support discovery and requires explicit device creation and management, or if the devices within this segment are always present and not subject to discovery. Always-present devices include virtual devices representing hardwired I/O, for example. auto Devices are automatically discovered and created in the hierarchy. This is suitable where crosstalk and ambiguity is not possible. For example, an edge server operating on a strictly private media may declare auto discovery mode.
manufacturer	string	(Optional) Reports the manufacturer. EXAMPLE "XYZ Corporation"
model	string	(Optional) Reports the model information. EXAMPLE "ABC 3000"
product	string	(Optional) Reports the product name. EXAMPLE "ABC by XYZ Corporation Smart Gateway"
version	string	(Optional) Reports the version of the product. EXAMPLE "2.08.034"
glpVersion	number	(Optional) The version of IAP. EXAMPLE "0"
capabilities	List	(Optional) Reports a list of capabilities for an edge server. Knowing these capabilities allows tools and the IAP management software to adjust its behavior accordingly. An edge server's capabilities do not change unless it is re-provisioned, repaired, replaced, or new firmware is loaded into it. The CMS considers the following capabilities unsupported unless they are included in the capabilities list: arbitrary-alarms Indicates that the edge server supports an alarm manager service that allows you to define conditions that raise and cancel

Field Name	Type	Description
		<p>alarms. See 5.2 Alarm.</p> <p>bridged-connections</p> <p>Indicates that the edge server supports point-to-point connections with infrastructure support. See 5.3 Connections.</p> <p>bulk-load</p> <p>Indicates that at least one protocol supported by this edge server supports the simultaneous loading of multiple instances of edge devices with the same type. See 5.13 Load Do Action.</p> <p>device-grouping</p> <p>Indicates that the edge server can take advantage of the device grouping feature. See the provision action in 5.5.4 Device Do Action.</p> <p>scheduler</p> <p>Indicates that the segment can manage schedules and scheduled events locally. When this capability is not declared, the CMS must manage the schedule within itself and activate the related jobs using a CMS-side scheduler. See 5.16 Schedule Services.</p> <p>history</p> <p>Indicates that a history of logged event records is supported and may be retrieved. See 5.8 Events.</p> <p>pagination-snapshots</p> <p>Indicates that pagination snapshots are supported for retrieval of logged event records. Resource-constrained edge servers may not be able to support snapshots. See 5.15 Query.</p> <p>lon-routing-rni</p> <p>Indicates support for EN 14908-1 remote network interfaces (RNIs).</p> <p>lon-routing-router</p> <p>Indicates support for EN 14908-1 routers.</p> <p>mixed-protocol-groups</p> <p>Indicates support for group objects with member lists that can contain devices, or items within devices, that use different drivers.</p>
limitations	Object	<p>(Optional) Reports limitations to features supported with this edge server. Each limitation is reported as a resource name/limit pair, with resource names exactly as listed below, and the corresponding limit value as discussed with each resource below.</p> <p>Some limitations are intrinsic and never change or change only under the same conditions which could also cause a change to capabilities (such as an edge server's capabilities, that change</p>

Field Name	Type	Description
		<p>whenever the controller is re-provisioned, repaired, replaced, or new firmware is loaded into it), some limitations may be configurable through a non-AIP interface and may change when the edge server resets.</p> <p>Limitations do not change without publication of the corresponding About object.</p> <p>The following limitations are defined. The limitations are in reference to IAP resources. For example, resources in reference to "groups" are in reference to "IAP group objects", which may or may not have a direct relationship to driver groups or groups exposed in the CMS.</p> <p>devices N (where N represents the number of edge devices allowed.)</p> <p>devices-per-group N (where N represents the number of edge devices allowed per group.)</p> <p>groups N (where N represents the number of groups allowed.)</p> <p>discoverDevices</p> <p>When discoverDevices is set to True, at least one of the drivers supports the discoverDevices action.</p> <p>When discoverDevices is either missing or set to False, then that device does not support discovery.</p> <p>See 5.7.1 Device Discovery.</p> <p>groups-per-device N</p> <p>N represents the number of groups that an edge device can be a member of.</p> <p>protocols Supported drivers specified by driver ID. The driver IDs listed in 4.4.2 Device Syntax are specified by this standard. Other driver IDs may be reported.</p>
languages	list	(Optional) List of supported languages for localized diagnostics.

Field Name	Type	Description																		
components	List of component objects	<p>(Optional) Edge servers report only components that can be individually updated or have been added, as might be the case with custom modules that you have added. Each component object provides the properties in the following table. The list of reported components may be empty, as might be the case with a monolithic edge server architecture.</p> <p>The component object properties:</p> <table> <tr> <th>Field Name</th><th>Type</th><th>Description</th></tr> <tr> <td>name</td><td>String</td><td> <p>(Required) The component's name.</p> <p>EXAMPLE</p> <p>"GLP Toolkit for Node.js"</p> </td></tr> <tr> <td>desc</td><td>String</td><td> <p>(Optional) Component's description.</p> <p>EXAMPLE</p> <p>"A toolkit of common services"</p> </td></tr> <tr> <td>license</td><td>String</td><td> <p>(Required) license reports a copyright notice, as shown, for proprietary software and licenses. Other components might report or refer to a different license, such as a URL to the MIT license if that component was governed by the MIT license.</p> <p>EXAMPLE</p> <p>"Copyright 2020 XYZ Inc."</p> </td></tr> <tr> <td>manufacturer</td><td>String</td><td> <p>(Required) The manufacturer's name.</p> <p>EXAMPLE</p> <p>"XYZ Inc. "</p> </td></tr> <tr> <td>uuid</td><td>String</td><td> <p>(Optional) uuid contains an optional component identifier. When provided, this must be a version 4 UUID, according to IETF RFC 4122. Compatible values are for example available from https://uuidgenerator.net uuid can be used by an app that relies on close communication with a specialized user interface component or any other case where a component must be uniquely identifiable by some other component.</p> <p>EXAMPLE</p> <p>"d193ab8c-bd98-4931-876b-23f5723861e5"</p> </td></tr> </table>	Field Name	Type	Description	name	String	<p>(Required) The component's name.</p> <p>EXAMPLE</p> <p>"GLP Toolkit for Node.js"</p>	desc	String	<p>(Optional) Component's description.</p> <p>EXAMPLE</p> <p>"A toolkit of common services"</p>	license	String	<p>(Required) license reports a copyright notice, as shown, for proprietary software and licenses. Other components might report or refer to a different license, such as a URL to the MIT license if that component was governed by the MIT license.</p> <p>EXAMPLE</p> <p>"Copyright 2020 XYZ Inc."</p>	manufacturer	String	<p>(Required) The manufacturer's name.</p> <p>EXAMPLE</p> <p>"XYZ Inc. "</p>	uuid	String	<p>(Optional) uuid contains an optional component identifier. When provided, this must be a version 4 UUID, according to IETF RFC 4122. Compatible values are for example available from https://uuidgenerator.net uuid can be used by an app that relies on close communication with a specialized user interface component or any other case where a component must be uniquely identifiable by some other component.</p> <p>EXAMPLE</p> <p>"d193ab8c-bd98-4931-876b-23f5723861e5"</p>
Field Name	Type	Description																		
name	String	<p>(Required) The component's name.</p> <p>EXAMPLE</p> <p>"GLP Toolkit for Node.js"</p>																		
desc	String	<p>(Optional) Component's description.</p> <p>EXAMPLE</p> <p>"A toolkit of common services"</p>																		
license	String	<p>(Required) license reports a copyright notice, as shown, for proprietary software and licenses. Other components might report or refer to a different license, such as a URL to the MIT license if that component was governed by the MIT license.</p> <p>EXAMPLE</p> <p>"Copyright 2020 XYZ Inc."</p>																		
manufacturer	String	<p>(Required) The manufacturer's name.</p> <p>EXAMPLE</p> <p>"XYZ Inc. "</p>																		
uuid	String	<p>(Optional) uuid contains an optional component identifier. When provided, this must be a version 4 UUID, according to IETF RFC 4122. Compatible values are for example available from https://uuidgenerator.net uuid can be used by an app that relies on close communication with a specialized user interface component or any other case where a component must be uniquely identifiable by some other component.</p> <p>EXAMPLE</p> <p>"d193ab8c-bd98-4931-876b-23f5723861e5"</p>																		

Field Name	Type	Description			
		<table><tr><td>version</td><td>String</td><td>(Optional) The version of the component. EXAMPLE "1.02.034"</td></tr></table>	version	String	(Optional) The version of the component. EXAMPLE "1.02.034"
version	String	(Optional) The version of the component. EXAMPLE "1.02.034"			
protocol	Protocol limitation Object	<p>The <i>protocol limitations object</i> contains limitation objects as described under limitations above, for each of the device protocols supported within the segment. This allows for the declaration of protocol-specific limitations.</p> <p>The global limitations are compiled from the individual protocol limitations. For example, the <code>global limitations.devices</code> property reports the total number of devices supported across all protocols while the <code>global limitations.devices-per-group</code> reports the minimum number of devices within a group supported by any of the supported protocols.</p> <p>Tools can explore global and protocol-specific limitations to prevent erroneous configurations and requests, but tools can also trust that any erroneous configuration or request yields an error.</p> <p>The limitations declared in this manner are guidelines only. For example, system or driver resource limitations may prevent the addition of a given device to a group, even though the group contains fewer then the declared <code>devices-per-group</code> limit.</p> <p>EXAMPLE</p> <pre>{lon: {devices: 32768, ...}, ... }</pre>			
mru	timestamp	The timestamp of the most recent update for this object. All feedback channel objects include the <code>mru</code> property to provide a timestamp of the most recent update for that object.			

5.3 Alarm

5.3.1 Definition

```
glp/0/{SID}/ (rq | fb) /alarm/{Handle}/{option}
```

The **alarm** topic allows you to define conditions that will trigger an alarm, as well as an optional action to be triggered by the alarm, such as publishing an alarm notification within IAP, sending an email or SMS notification, or executing a job object.

IAP includes an alarm manager service that monitors IAP for all alarm conditions and, when the alarm conditions are observed, it triggers the event reporting and logging services, as well as any alarm actions that are defined for the alarm. See **5.2.7 Alarm Actions**.

5.3.2 Example

A user could define an alarm to be triggered whenever a particular datapoint reports power consumption that exceeds 750 watts, and to send an email to a technician when the alarm is triggered. When the IAP alarm manager detects the alarm condition to be true, the alarm manager uses the event reporting and logging services to report the event, and also triggers the action defined for the alarm (sending an email to a technician).

The triggered actions can be configured, and their messages can be customized using a selection of macros. These macros are described under the `message` property in Table 13 below.

IAP identifies four states for alarms, and whenever an alarm changes state, the alarm manager reports the change through the event and event logging services. Table 11 list the alarm states.

Table 11 — Alarm States

Active	An alarm is triggered because the conditions defined for an alarm have been met.
Inactive	An alarm is cancelled because either the conditions to cancel an alarm have been met, or the conditions defined to trigger the alarm are no longer being met.
Acknowledged	A human operator responds to an alarm by performing an action through the CMS (such as clearing the alarm). Whether an alarm requires acknowledgment is configurable. An alarm that does not require explicit acknowledgment is always considered acknowledged.
Unacknowledged	A triggered alarm that is configured to require acknowledgment, has not been acknowledged by a human operator.

The alarm's state (active or inactive, and acknowledged or unacknowledged) is reflected in the events channel for the selected category. See **5.8 Events**.

Multiple conditions can be defined for an alarm. An alarm can become active when one condition is met, such as a power reading exceeding 750 W, and can return to an inactive state when another condition is met (such as the power reading dropping back below 500 W). Setting a second condition is optional; if none is provided, the one condition determines when the alarm becomes active and when it returns to inactive. In this case, with one condition set, the alarm would become inactive when the power reading drops below 750 W.

Deactivating an alarm never requires acknowledgment. But for an alarm that requires explicit acknowledgement, the alarm does not clear until it is acknowledged. An alarm that does not require explicit acknowledgment is always considered acknowledged, therefore, it is automatically cleared once the deactivation condition is met.

Alarm Lifecycle Example:

Consider an alarm that activates when a temperature reading exceeds 80 °C, deactivates once the temperature falls below 70 °C, and is configured to require explicit acknowledgment.

- 1) When the temperature exceeds 80 °C, the alarm becomes active and in an unacknowledged state, awaiting acknowledgment. The alarm manager begins tracking this alarm and reports the change in alarm state to the event and logging service.
- 2) When the temperature drops below 70 °C, the alarm becomes inactive, but remains unacknowledged. This new state is reported to the event and logging service.

When the CMS submits explicit acknowledgment of the alarm, it remains inactive, but becomes acknowledged. This new state is reported to the event and logging service. The alarm manager stops tracking the alarm.

5.3.3 Alarm Configuration

The alarm’s current configuration is reported in the following topic with the properties listed in the Alarm Properties table described in **5.2.6 Alarm Properties**.

```
glp/0/{SID}/fb/alarm/{Handle}/cfg
```

5.3.4 Alarm Status

The alarm definition status is reported in the following topic:

```
glp/0/{SID}/fb/alarm/{Handle}/sts
```

Alarm status is reported with the **state** property described in Table 12.

Table 12 — Alarm Status Property

Field Name	Type	Description
state	string	Possible values: "enabled", "disabled", or "deleted".

5.3.5 Examples

Example of configuration object published on glp/0/lsg/fb/alarm/k/cfg

```
{
  "desc": null,
  "enabled": true,
  "cat": "test",
  "ackd": true,
  "activation": "gt(\"glp/0/lsg/fb/dev/lon/0/if/light/0/sp/value/h\", 10) ",
  "deactivation": "eq(\"glp/0/lsg/fb/dev/lon/0/if/light/0/sp/value/h\",
  30) ",
  "message": null,
  "urgent": false,
  "actions": null,
  "severity": null,
  "mru": "2019-03-22 06:29:05.774 UTC"
}
```

Example of status object published on glp/0/lsg/fb/alarm/k/sts

```
{
  "state": "enabled",
  "mru": "2019-03-22 06:29:03.845 UTC"
}
```

5.3.6 Alarm Properties

Each alarm supports the properties listed in Table 13.

Table 13 — Alarm Properties

Field Name	Type	Description
desc	String	(Optional) A description of the alarm.
enabled	Boolean	(Required) Indicates whether the alarm is enabled.
cat	String	(Optional) Alarm category to use when the alarm is reported through the event and event logging service. The available categories are <code>alarm</code> and <code>data</code> . You can also define your own category. The default is <code>alarm</code> .
ackd	Boolean	(Optional) Defaults to True.
activation	String	(Required) Alarm condition expression to activate the alarm. See 5.2.8 Alarm Conditions .
deactivation	String	(Optional) Alarm condition expression to deactivate the alarm. See 5.2.8 Alarm Conditions .
message	String	(Optional) Alarm message, defaults to the activation alarm condition. The alarm message is used when reporting the alarm to the event and logging service. It is also sent with SMS actions (clipped when necessary) and used for the body of an email action. The message can be configured using a simple template language

Field Name	Type	Description										
		with placeholders enclosed in braces:										
		<table><tr><td>{topic}</td><td>Feedback topic under scrutiny, according to the alarm condition.</td></tr><tr><td>{alarm}</td><td>Alarm’s feedback topic.</td></tr><tr><td>{condition}</td><td>Condition that triggered the activation.</td></tr><tr><td>{date}, {time}</td><td>Current date and time.</td></tr><tr><td>{sid}</td><td>Segment identifier.</td></tr></table>	{topic}	Feedback topic under scrutiny, according to the alarm condition.	{alarm}	Alarm’s feedback topic.	{condition}	Condition that triggered the activation.	{date}, {time}	Current date and time.	{sid}	Segment identifier.
		{topic}	Feedback topic under scrutiny, according to the alarm condition.									
		{alarm}	Alarm’s feedback topic.									
		{condition}	Condition that triggered the activation.									
		{date}, {time}	Current date and time.									
		{sid}	Segment identifier.									
The default value is "alarm".												
urgent	Boolean	(Optional) Alarm urgency, defaults to False. The alarm urgency is reported to the event reporting service. See 5.8 Events .										
actions	Object	(Optional) List of actions. Possible values: "sms", "email", and "job". See 5.2.7 Alarm Actions . EXAMPLE <pre>{sms: { ... }, email: { ... } ...}</pre>										
severity	String	(Optional) string describing the alarm severity in user-defined categories, such as: — "minor fault" — "major fault" — "critical fault" The string can be up to 60 characters. The severity, if available, is included with the Event object that reports the alarm. See 5.8.5 Event Object Properties .										

5.3.7 Alarm Actions

An alarm manager can invoke actions when an alarm activates. There are three basic Alarm actions: sms, email, and job. Each action is identified with its keyword and one or more action-specific attributes. Multiple actions can be configured for an alarm. All configured actions occur when the alarm activates.

Specific implementations may support other alarm action types, such as Slack or Twitter notifications. Actions include template text which can be parameterized with several macros, such as the alarm source, and the condition which was met to trigger the alarm. The available actions are listed in each product's documentation.

Table 14 lists the basic Alarm actions.

Table 14 — Alarm Actions

Action	Description
sms	Sends an SMS text message to the specified cell phone. The sms action has one property: to, which contains an email address for use with an Email-to-SMS Gateway service, or a comma-separated list of such addresses to send the alarm to. Because SMS

Action	Description
	<p>messages are sent using Email-to-SMS gateway services, a correct SMTP service configuration is required. See the <code>SMTP</code> property in the Segment Configuration Properties table described in 5.17.2 Segment Configuration Properties.</p> <p>The SMS message body contains the alarm message (see the <code>message</code> property in the Alarm Properties table described in 5.2.5 Alarm Properties), truncated to fit the SMS format, if necessary. SMS services require that the edge server has Internet access.</p> <p>EXAMPLE</p> <pre>sms: { to: "4081234567@sms.mycellco.com" }</pre>
email	<p>Sends an email message to the specified email account. The <code>email</code> action has two properties:</p> <ul style="list-style-type: none"> — <code>to</code>, which accepts a comma-separated list of destination email addresses. — <code>subject</code>, which contains the subject for the message. <p>The message body is constructed from the alarm message (described in the <code>message</code> property in the Alarm Properties table described in 5.2.6 Alarm Properties). The same macros that are available for the message body can also be used for the message subject.</p> <p>Emails are sent through the SMTP server configured with the segment (see the <code>SMTP</code> property in the Segment Configuration Properties table described in 5.17.2 Segment Configuration Properties). Email services require that the edge server has Internet access.</p> <p>EXAMPLE</p> <pre>email: { to: "supervisor@xyz.com, abc@gmail.com", subject: "Alarm: {condition}" }</pre>
job	<p>Runs the specified job. The <code>job</code> action has three properties:</p> <ul style="list-style-type: none"> — <code>activate</code> — <code>deactivate</code> — <code>acknowledge</code> <p>Each property can specify an IAP job to be executed when the alarm is activated, deactivated, or acknowledged, respectively.</p> <p>Jobs are referenced with their request channel topic.</p> <p>EXAMPLE</p> <pre>job: { activate: "glp/0/downtown/rq/job/123" }</pre>

5.3.8 Alarm Conditions

5.3.8.1 Definition

```
eq(topic, value)
gt(topic, value)
lt(topic, value)

updated(topic, interval, unit)
changed(topic, interval, unit)
changed(topic, interval, unit, threshold)

recent(topic)
```

IAP provides a number of functions that allow you to establish the desired conditions for your alarms. The alarm condition is expressed in JavaScript. Any valid JavaScript construct may be used to combine or evaluate the results of these functions, or to construct arguments to these functions. The *topic* is typically, but not necessarily, a feedback channel topic.

5.3.8.2 Comparison Functions

```
eq(topic, value)
gt(topic, value)
lt(topic, value)
```

There are three comparison functions `eq` (equals), `gt` (greater than), and `lt` (less than). These functions compare the most recent value received for the specified *topic* with the specified *value*, and return a Boolean result. The *topic* is typically, but not necessarily, a feedback channel topic.

The alarm condition is expressed in JavaScript. JavaScript expressions that use the standard JavaScript operators can be used to combine or evaluate the results of these functions, or to construct arguments to these functions.

When the alarm manager encounters the *topic* for the first time, it separates it into the IAP resource *topic* and the property selector parts, and subscribes to the IAP resource *topic* to track changes. When a newly published value is received on that topic, all alarm conditions that are based on that topic are evaluated. A resource topic is a topic that appears in one of the IAP channels, and the property selector is a slash-separated list of properties within that topic.

Topic examples:

This topic refers to an IAP resource:

```
glp/0/dt/fb/dev/cm/0/if/light/0
```

This topic refers to a property within the same IAP resource:

```
glp/0/dt/fb/dev/cm/0/if/light/0/sp/value/b
```

In the three comparison functions (`eq`, `gt`, `lt`), *value* accepts any value compatible with the *topic*: a string, number, Boolean constant, list, or object.

— Null can only be equal to itself (null `eq` null is always True).