

The Lucid Protocol specification

Version: 3.1
Date: 16-June-2026

Disclaimer and license agreement

DISCLAIMER	**Whilst all due care has been taken to prepare this material, the accuracy and completeness of any information cannot be guaranteed, and you rely upon and or use the material at your own risk.**

By using and/or copying this document, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions:

Permission to copy, and distribute the contents of this document, in any medium for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the document or portions thereof:

- A link or URL to the original WITS Protocol Standards Association document.
- All existing copyright notices, or if one does not exist, a notice (hypertext is preferred, but a textual representation is permitted) of the form: "Copyright © 2019-2026 WITS Protocol Standards Association".

Inclusion of the full text of the Copyright content (see below) must be provided. Authorship attribution must be provided in any software, documents, or other items or products that you create pursuant to the use of the contents of this document, or any portion thereof.

No right to create modifications or derivatives of WITS Protocol Standards Association documents is granted pursuant to this license, except anyone may prepare and distribute derivative works and portions of this document in software that implements the specification, in supporting materials accompanying such software, and in documentation of such software, provided that all such works include the notice below. However, the publication of derivative works of this document for use as a technical specification is expressly prohibited.

As an exception to the above, anyone may fork the repository containing this document and make temporary modifications within that fork, for the purpose of proposing contributions back to the Lucid Protocol project. This does not permit the independent publication of derivative specifications, which remains prohibited.

Copyright

Copyright © 2019-2026 WITS Protocol Standards Association. This software or document includes material copied from or derived from the Lucid Specification.

THIS DOCUMENT IS PROVIDED "AS IS" AND IS USED AT YOUR OWN RISK. THE COPYRIGHT HOLDERS AND THE WITS PROTOCOL STANDARDS ASSOCIATION MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD-PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER INTELLECTUAL PROPERTY RIGHTS. TO THE MAXIMUM EXTENT PERMITTED BY LAW, THE COPYRIGHT HOLDERS AND THE WITS PROTOCOL STANDARDS ASSOCIATION WILL NOT BE LIABLE TO YOU OR ANY THIRD PARTY WHETHER IN CONTRACT, TORT, NEGLIGENCE, BREACH OF STATUTORY DUTY OR OTHERWISE, FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL LOSS OR DAMAGES OF ANY KIND, LOSS OF PROFIT, LOSS OR DAMAGE TO REPUTATION OR GOODWILL, LOSS OR CORRUPTION OF DATA, LOSS OF BUSINESS OR LOSS OF REVENUE ARISING HOWSOEVER OUT OF ANY USE OF THIS DOCUMENT OR THE PERFORMANCE OR IMPLEMENTATION OF THE CONTENTS THEREOF.

The name and trademarks of the copyright holders or the WITS Protocol Standards Association may NOT be used in advertising or publicity pertaining to this document or its contents without specific, written prior permission. Title to copyright in this document will at all times remain with copyright holders. The information contained herein is subject to change without notice.

Contact address

Website: http://www.lucidprotocol.org/

Send any enquiries to: enquiries@lucidprotocol.org

Notices

The following safety notices are used as standard throughout the documentation. Make sure you fully understand and comply with the notices in this manual.

NOTICE	**Indicates an important situation which, if not avoided, may seriously impair operations.**

NOTE:	**Additional information relating to the current section.**

Conventions

This doc uses the following conventions:

Code listing:

```
  "lucid": 3, // Protocol version
```

The key words ”MUST”, ”MUST NOT”, ”REQUIRED”, ”SHALL”, ”SHALL NOT”, ”SHOULD”, ”SHOULD NOT”, ”RECOMMENDED”, ”MAY”, and ”OPTIONAL” in this document are to be interpreted as described in RFC 2119.

Glossary of terms

The following abbreviations and terms are used throughout this document.

Abbreviation/Term	Definition
BOM	Byte Order Mark
Broker	Also named MQTT Server. Service for storing and forwarding MQTT messages.
Device Configuration	JSON structure, following a defined format, that expresses how the capabilities in the Device Profile have been deployed in the FD.
Device Profile	JSON structure, following a defined format, that describes the capabilities of a Field Device.
FD	Field Device, also known as Outstation or Remote Terminal Unit (RTU). Equipment located near an asset that needs to be monitored and/or controlled.
Field Device Link	A channel which enables the Supervisory Application to access the Field Device
IIOT	Industrial Internet of Things
IOT	Internet of Things
JSON	Javascript Object Notation. Open standard format for human readable data interchange.
LWT	Last Will and Testament. A message sent from the Broker to subscribers when the publishing device is involuntarily disconnected from the Broker.
MQTT	Lightweight publish/subscribe (pub-sub) machine-to-machine network protocol.
OT	Operational Technology

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

Abbreviation/Term	Definition
Payload	Data items represented in the protocol in JSON format.
Point	Addressable, physical, or virtual input or output on an FD. Also known as Channel, Tag, Signal, or Metric.
QoS	MQTT Topic Quality of Service
SA	Supervisory Application, also known as Master Station (MS). Service for acquiring and processing data, monitoring and controlling devices and processes.
Topic	MQTT name for a message stream on an MQTT Broker, notionally similar to a folder structure on a PC. MQTT applications publish and subscribe to Topics to exchange messages.
UUID	Universally Unique IDentifier is a 128-bit (16 octet) label used for identifying components in computer systems. When generated according to the standard methods, UUIDs are unique, recorded as a lower case string of 36 characters following a standard format defined in section 4 of RFC9562, for example: 123e4567-e89b-12d3-a456-426614174000.

References

[JSON]	JSON: https://www.ietf.org/rfc/rfc4627.txt
[MQTT]	MQTT Version 3.1.1. 2014. OASIS Standard: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html MQTT Version 5. 2019. OASIS Standard: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html
[JSONSchema]	JSON Schema: http://json-schema.org/
[RFC2119]	RFC2119 Key words for use in RFCs to Indicate Requirement Levels: https://datatracker.ietf.org/doc/html/rfc2119
[Timestamps]	RFC3339 Timestamps: https://tools.ietf.org/html/rfc3339
[BASE64]	RFC4648 The Base16, Base32, and Base64 Data Encodings: https://datatracker.ietf.org/doc/html/rfc4648
[MIME]	RFC1341 Multipurpose Internet Mail Extensions: https://datatracker.ietf.org/doc/html/rfc1341
[GZIP]	GZIP data compression: https://datatracker.ietf.org/doc/html/rfc1952
[Brotli]	Brotli data compression: https://datatracker.ietf.org/doc/html/rfc7932
[Deflate]	Deflate data compression: https://datatracker.ietf.org/doc/html/rfc1951
[UUID]	Universally Unique Identifiers: https://datatracker.ietf.org/doc/html/rfc9562

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

Table of contents

Disclaimer and license agreement 1

Copyright 1

Contact address 2

Notices 2

Conventions 2

Glossary of terms 2

References 3

1 Introduction 6
- 1.1 Purpose . . . 6
- 1.2 Use of the protocol . . . 6
- 1.3 Protocol history . . . 7
- 1.4 Protocol transport . . . 7
- 1.5 Supervisory application and field device relationship . . . 7
- 1.6 Lucid between supervisory applications . . . 7
- 1.7 Data representation . . . 8

2 Data model 9
- 2.1 Field device . . . 9
  - 2.1.1 Lucid schema . . . 11
  - 2.1.2 Device profile . . . 11
  - 2.1.3 Configuration . . . 15
  - 2.1.4 Connection scheduling . . . 19
  - 2.1.5 Time formats . . . 20
- 2.2 Events . . . 20
- 2.3 Actions . . . 20
- 2.4 Points . . . 21
  - 2.4.1 Point quality flags . . . 25
  - 2.4.2 Binary points . . . 26
  - 2.4.3 Analogue points . . . 29
  - 2.4.4 Counter points . . . 32
  - 2.4.5 Analogue limits . . . 33
  - 2.4.6 Counter limits . . . 40
  - 2.4.7 No change . . . 43
- 2.5 Extended points . . . 44
  - 2.5.1 Minimum . . . 45
  - 2.5.2 Maximum . . . 46
  - 2.5.3 Mean . . . 46
  - 2.5.4 Standard deviation . . . 46
  - 2.5.5 Integral . . . 46
  - 2.5.6 Rate of change . . . 47
  - 2.5.7 State counter . . . 48
  - 2.5.8 State runtime . . . 49
- 2.6 Limit and control profiles . . . 49
  - 2.6.1 Limit profiles . . . 49
  - 2.6.2 Control profiles . . . 50
- 2.7 Calculated points . . . 50
- 2.8 Incident logging . . . 51

3 Controls and overrides 52

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

3.1 Controls . . . 52
3.2 Overriding a point . . . 52

4 Data reporting **54**
4.1 Data formats . . . 54
4.2 Requesting data . . . 55

5 Field device core blocks **57**

6 Use of MQTT **58**
6.1 Retained messages and connection state . . . 58
6.2 Publish/subscribe transports . . . 58
6.3 Payload . . . 58
6.3.1 Payload compression . . . 58
6.4 Other topics . . . 59
6.4.1 Getdeviceinfo . . . 59
6.4.2 Maintaining time . . . 59
6.5 Communications, QoS, retention . . . 60

7 Field device lifecycle **61**
7.1 Status message . . . 61
7.2 Last Will and Testament . . . 63
7.3 Field device replacement . . . 63

8 Security **65**

9 Class names and ontology **67**
9.1 Fixed class names . . . 67

10 Field device profile extensions and layout **69**
10.1 Vendor extensions . . . 69
10.1.1 New properties . . . 69
10.1.2 New objects . . . 69
10.1.3 Translation . . . 69
10.2 Suggesting SA UI layout . . . 69
10.2.1 configFields . . . 69
10.2.2 Group . . . 69

11 Example use cases **71**
11.1 Simplest case . . . 71
11.2 A configurable field device . . . 71

12 Supervisory application profile document **73**

A Appendix: JSON files **75**

B Appendix: Topic summary **76**

C Appendix: Calculated point code **77**
C.1 Evaluation of a calculated point value . . . 77
C.2 Operation codes . . . 78
C.3 Example . . . 81
C.4 Quality flags . . . 81

D Appendix: Revision history **83**

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

1 Introduction

This document provides a comprehensive guide to the Lucid Protocol (Lucid), a standard that permits electronic devices deployed at user sites to communicate with server systems within the user's offices. The telemetry data sent using Lucid permits operational staff working for the user to visualize and control their system.

Lucid provides a rich set of functions, previously implemented in telemetry and remote SCADA systems, using manufacturer-specific protocols. Additionally, it allows for interoperability between equipment from different manufacturers. By implementing a common standard, Lucid Protocol users can purchase and operate data sources (Field Devices) and server software (Supervisory Applications) from multiple vendors and be assured that they are compatible. This encourages competition between vendors and reduces the costs of procurement, training, operations, and maintenance.

This document contains the technical detail of the protocol. Appendices contain protocol schema and samples.

1.1 Purpose

The protocol can be used in any industry for the management of remote data gathering and control devices.

Goals for Lucid include:

- Is simple to implement.
  Built on standard protocols, techniques, and libraries.
- Makes efficient use of network resources.
  Needs to work in all common situations from application to application transferring thousands of packets per minute, down to tiny devices using unreliable slow connections.
- Has a flexible architecture.
  Connects devices to single or multiple applications and a wide variety of communications scenarios.
- Maintains common outstation and data logging functionality.
  Has the smallest cost to implement and shares the philosophy and terminology of earlier WITS systems.
- Economic to maintain validation.
  Simple to test and assure functionality and compatibility.
- Security.
  Security features are included in the protocol.

1.2 Use of the protocol

Lucid is used to provide communications between a remotely deployed Field Device (FD) and one or more central Supervisory Applications (SA), and between Supervisory Applications.

NOTE:	**The communications between FD to SA go through an intermediate Broker. So, whenever the text describes communications from an FD to an SA, or an SA to an FD, the Broker should be inferred.**

Lucid supports the transmission of information from the FD to the SA. That information is generally related to points on the FD that provide values users are interested in collecting or monitoring.

The FDs may be constrained devices of a similar nature to those deployed in Internet of Things (IoT) and Industrial Internet of Things (IIoT), or as smaller devices within Operational Technology (OT), but Lucid is not limited to these styles of devices. Indeed, Lucid may be used between SAs.

Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

The types of functions supported by Lucid are those which from normal IoT and OT operations. The functions originally rose out of requirements within the UK Water Industry and are now considered as common functions across the whole of industrial IoT. Lucid supports the following functions:

- Point information.
- Point change reporting.
- Control of point values.
- Limit and Control Profiles to be used in reporting change or controlling point values.
- Time-summarised points.
- Calculated points.
- Logging of point data for later retrieval.
- Synchronisation of time.
- Management of connections.
- Definition of device capabilities.
- Monitoring and controlling device configuration.
- Provision of security.
- Vendor customisable extensions for SA UI layout.
- Download and upgrade of device firmware.

1.3 Protocol history

Lucid was inspired by the WITS-DNP3 protocol, which is deployed throughout the UK, largely by water companies. It was first named WITS-IoT and was rebranded to Lucid in early 2023.

1.4 Protocol transport

Lucid uses MQTT as a transport mechanism. Messages are exchanged over several predefined Topics in an MQTT Broker.

For more information, see Section 6, Use of MQTT.

Lucid users publish and subscribe to these Topics to exchange messages of defined formats.

Lucid formalises the definition of the FD and SA and how they are expected to operate to allow specification of the message content. Lucid provides a definition of the JSON to be used as the content of the MQTT messages. This is provided through examples and the use of a JSON schema.

For more information, see [JSONSchema].

1.5 Supervisory application and field device relationship

There is a many-to-many relationship between SAs and FDs. That is, an FD may report and be controlled by multiple SAs via the Broker. Also, an SA may monitor and control many FDs. The exchange of a Configuration Version (cVer) between FDs and SAs provides the mechanism to manage change.

1.6 Lucid between supervisory applications

Lucid may be used between SAs, without an FD. In this operating mode, one SA assumes the role of an FD and generates Lucid Protocol content as if it was a Lucid FD. This can be useful for sharing FD configuration, data, and control messages between Lucid SAs.

Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

Example:
This may be used where a Lucid SA also operates non-Lucid native FDs and needs to share the configuration, data, and control messages with a separate Lucid SA. This allows a pure Lucid SA to fully operate with non-Lucid FDs using an intermediary Lucid-to-Native SA.

1.7 Data representation

Lucid consists of MQTT Topics and JSON Payloads:

- A Topic defines a named path to data.
- A Payload could be any data items represented in Lucid in JSON format but may be compressed as specified by the Payload content.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

2 Data model

Communications are defined between an SA and FDs that can use any software or hardware architecture. Multiple SAs and FDs are possible on the same system.

2.1 Field device

A Field Device is a piece of equipment that is monitoring or controlling one or more physical assets. Each asset may have one or more attributes that the FD is monitoring or controlling.

Examples of this are:

- A pressure sensor on a pipe monitoring the pressure in the pipe.
- A sensor monitoring a water meter recording water flow over time.
- An FD controlling the switch on a water pump.

FDs have sensor inputs and outputs called points that can be individually configured, read, and recorded. Points each have a set of configuration data, often with a wide array of attributes, describing the capabilities of the point or asset being monitored and controlled.

Changes of point values, states and other things can occur within the FD. The protocol allows actions to be associated with these changes on the FD. The actions may cause the generation of events so that the SA can be informed of the changes. Typical FD changes can be for example:

- A monitored point may record a value that is greater than a predefined limit.
- An internal timer in the FD may trigger.
- An external sensor may become inoperative.

FD actions may include recording the issue or event or triggering a communications operation.

Every FD MUST have a Universally Unique Identifier (UUID). This is so that every Lucid device can be uniquely identified, regardless of communications medium or site location. The UUID MUST be initially set with a globally unique string by a device vendor.

Figure 1 shows a series of messages between an FD and an SA, with a Broker as intermediary.

For more information, see Section 11, Example use cases for sample message content.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

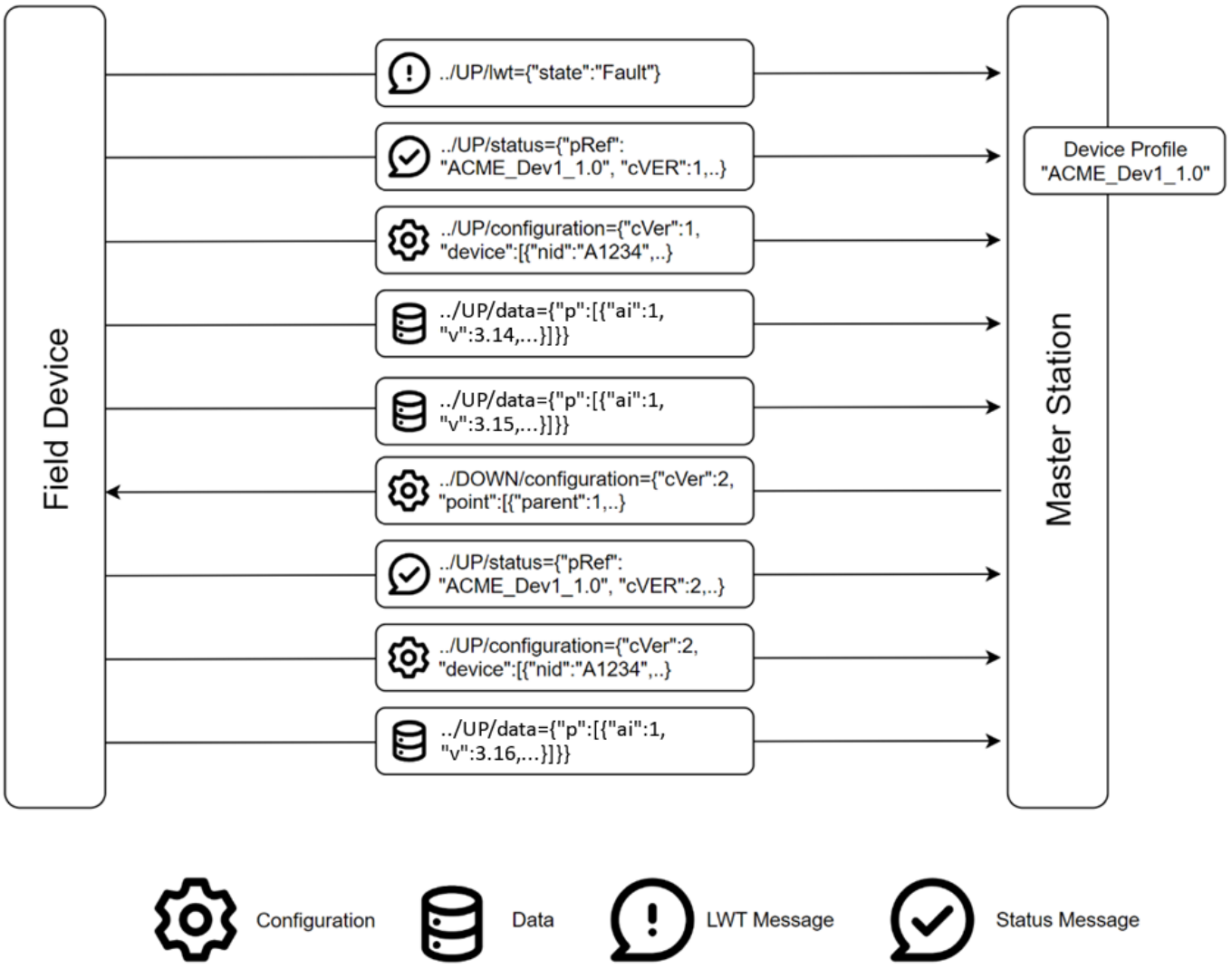


Figure 1: Field device and supervisory application messages

The FD creates a connection, delivering an LWT message to the Broker, which holds that message for delivery to the SA in the event of unexpected disconnection of the FD.

For more information, see Section 6, Use of MQTT which describes session and online status.

The FD sends status and configuration messages to the SA, enabling the SA to set up the device signals, and properties, in its database. To do this it refers to the FD Profile for that FD type. The FD profile and configuration message may optionally be sent from FD to SA. Where the FD profile and/or configuration message is not sent by the FD, it could be sent by a third party to the same Topic.

By omitting the FD Profile and configuration messages, lower device bandwidth requirements can be achieved.

The FD sends data messages to the SA. These are unsolicited and controlled by configuration properties and FD behaviour. The FD has a queue of data items to be sent via the Broker. There is a Topic and Payload available for the SA to request data, and such data requests are also queued.

The FD delivers all queued data to the SA via the Broker, when a connection is established, controlled by schedules, and defined actions. While data may be marked by such event priorities, all data is delivered in the queued order.

The SA can update FD configuration by sending some or all configuration properties, as defined by the FD Profile. These result in a new status message from the FD and a new full configuration message, confirming the changes. A property of the configuration message states whether the message contains the complete configuration or just part.

Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

2.1.1 Lucid schema

The content of all Lucid messages is laid out according to a schema using the JSON Schema format as shown in the JSON code file: **lucid.schema.json**.

This schema defines the necessary fields for each type of message, In the case of the objects and fields in the profile, the schema defines the structure but not each individual field.

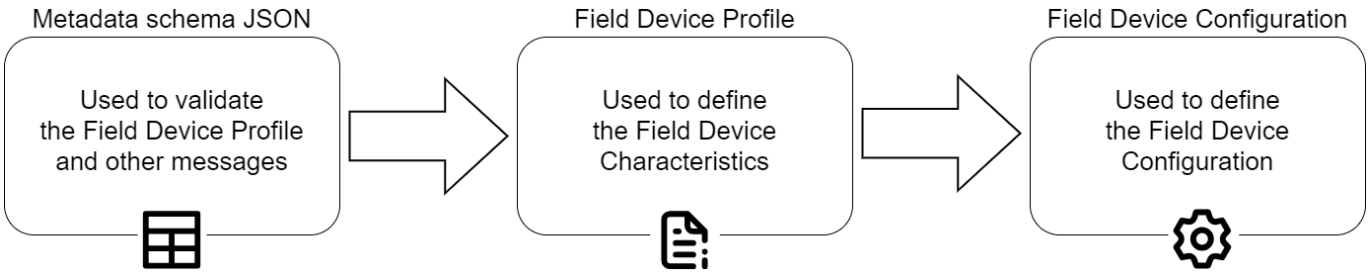


Figure 2: JSON schema

2.1.2 Device profile

The FD Profile describes the capability of the device, and how and what can be configured. It is a JSON document that can be used by the SA to determine what configuration can be used with the specific FD.

The FD Profile specifies the available upload/download characteristics, the device features, and the allowable properties for each element of the configuration. It also contains hardware version and manufacturer information.

The FD Profile can either be sent from the device or referred to by a unique reference (preferred) where the profile has previously been uploaded to the SA. The FD Profile is pre-loaded into the SA, as there can be many FDs sharing the same profile.

The FD sends the Device Profile, or reference, on first connection to the MQTT Broker and afterwards on request. First connection is determined by the FD.

If referred to by a unique reference, the SA retains the Device Profile and creates or interprets FD configuration messages accordingly.

The unique reference is a string containing the manufacturer name, device (product) name and the version number, each separated by underscores and with no full stops. When an FD Profile reference is received by the SA, it can find the FD Profile from preloaded files.

The JSON code file, **Field Device Profile.json,** supplied with the Lucid specification defines a typical fully featured FD Profile. Not all features of this FD Profile are required by an FD; types and fields can be omitted. The specific Device Profile for an FD describes all the features available on the device, relevant to its type, and the capability (capacity) of its signals and communications. Only properties listed in the FD Profile may be included in the FD configuration.

The FD Profile Topic name is:

[UUID]/UP/profile

The QoS is 1, RETAIN is 1.

2.1.2.1 Customising the field device profile

The **Field Device Profile.json** file, supplied with the Lucid specification, can be modified for an FD implementation as follows:

- Change the Profile Reference pRef field to uniquely identify the profile.
- Change or remove parts of the capability section which describes the supported commands, compression, and quantities of items, and specifically named items such as channels or device links.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

- Remove any complete object type definitions from the objectTypes section.
  For example, an FD may not support counters or alarm limit profiles, so these sections are omitted.
- Remove any configuration fields from the configFields section.
  For example, an FD may not support chatter detection on points, so the whenChatter field is omitted.
- Remove any extension point types (or the whole extension point scheme) if not supported.
- Add any product-specific objects and tables using a manufacturer-specific naming scheme.

The following sections describe each part of the FD Profile.

2.1.2.2 Field device profile: field device version information

This lists the Lucid version, product name and manufacturer version, and the capabilities for upload and download of core and configuration.

```
  "lucid": 3,              // Protocol version
  "pRef": "ACME_Dev1_1-1", // These three fields uniquely define the device
                           // profile. By convention the profile name is the
                           // manufacturer name, device name and version
                           // identification in that order, separated by
                           // underscores. E.g. " ACME_Dev1_1-1". This is used in
                           // the status message. Avoid using periods or other
                           // characters not commonly used in filenames.
  "canCoreUp": false,      // Whether Core files can be uploaded/downloaded
  "canCoreDown": false,
  "canConfigDown": true,   // Whether the device can receive configuration
```

The Device Profile is then divided into a Capability section and an Object Types section. These define the general capabilities of the FD, and the quantities and addressing for a collection of objects which make up the FD configuration.

The remainder of the Device Profile is arranged to allow the SA to create the user interface and database structures, to store and edit or view configuration, and to upload or download the resulting configuration derived from the Device Profile.

The Device Profile is also designed to allow manufacturer-specific fields to be identified, so that new, device specific features can be added. It also allows extensibility of the Device Profile standard features without having to create new SA software to support FD features.

2.1.2.3 Field device profile section: field device capability

These items are present to inform the SA how many objects of each type can be configured and how they are to be addressed.

Points

- An array of objects indicating each point type and quantity.
- The list of points denotes physical capability, but not all points need to be configured or used.
- Points may be wired interfaces to other devices or virtual in software. This is device specific.
- An array of types and numbers is used. It is verbose, but simple and flexible.
- Numbers can be 0 or any positive integer.
- Number ranges across types are not unique, so that, for example, there can be analogue input 1 and analogue output 1.
- Point IDs are unique across device links. That is, you cannot have AI0 in two different device links. Device links are just collections of points.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

For example:

```
"body": {
  "capability": {
    "points": {
      "ai": [
        {
          "from": 1,
          "to": 2
        },
        {
          "from": 6,
          "to": 8
        }
      ],
```

This defines the available ranges for analogue input points: 1, 2, 6, 7 and 8.

Channels

This lists the channel numbers available for a named channel. Numbers can be 0 or any positive integer. Each channel represents a single physical interface by which the Broker can be reached. As an example, the following shows a device which has an ethernet and mobile channel defined. The name and implementation of a channel are device specific. The name is expected to be indicative of the channel's nature. Each channel can support multiple connections.

```
    "channels": [
      {
        "name": "Ethernet 1",
        "num": 1
      },
      {
        "name": "GPRS",
        "num": 2
      }
    ],
```

The FD only ever communicates with a single Broker system. The exact configuration of the Broker system is up to the user. Multiple channels and connections allow some redundancy in the method of connection to the single Broker system. As there is only a single Broker system, the FD need only maintain a single store of messages for that Broker system.

Once messages are available at the Broker, multiple SAs can subscribe to the messages and receive them.

Field device links

A Field Device Link enables an FD to organise its points into sets. Typically, these sets share the same method of access. For example, there may be physical I/O on the FD, plus a separate capability to link to a Modbus I/O connection or a rack of I/O cards. This section names and assigns numbers to the FD Links. Device Link names can be any text. Numbers can be 0 or any positive integer.

```
    "deviceLinks": [
      {
        "name": "Internal",
        "num": 0
      },
      {
        "name": "Onboard",
        "num": 1
      },
```

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

```
      {
        "name": "PLC1",
        "num": 2
      },
      {
        "name": "Calc",
        "num": 3
      }
    ],
```

Commands

This denotes what the SA can send to perform actions on Devices with this Device Profile. For example, getdata (manual retrieval of data) and setclock (SA set of clock). If the SA sends these request types, then the FD will respond.

```
    "commands": [
      "getdata",
      "setclock",
      "control",
      "override",
      "getdeviceinfo"
    ]
```

2.1.2.4 Field device profile section: object types

The Object Types section of the FD Profile details objects and their fields which can be configured within the configuration messages. They define how configuration messages are formed.

```
"objectTypes": [
```

For further information about forming the Configuration message, see Section 2.1.3, Configuration.

The range of items permitted within the Field Device Profile is described in the JSON schema document, enabling the SA to self-configure its database schema for new features. To do this the protocol uses some commonly used conventions in JSON libraries for consistency, for example a form library http://schemaform.io and JSON schema definitions in http://json-schema.org.

The object types are listed in the profile as a series of arrays. However, their meaning is explained as a hierarchy, and within each object there are references to related objects. Figure 3 shows the structure of the object types, with the relationships and cardinality between objects. That is, (1) means 'one of' and (0..n) means 'zero or more'.

NOTICE	**These numbers do not refer to the index numbers of objects. The index numbers are the values stated in the Device Profile's 'body'/'capability' section in the "from" and "to" parameters. See section 2.1.32.1.2.3, Field device profile section: field device capability.**

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

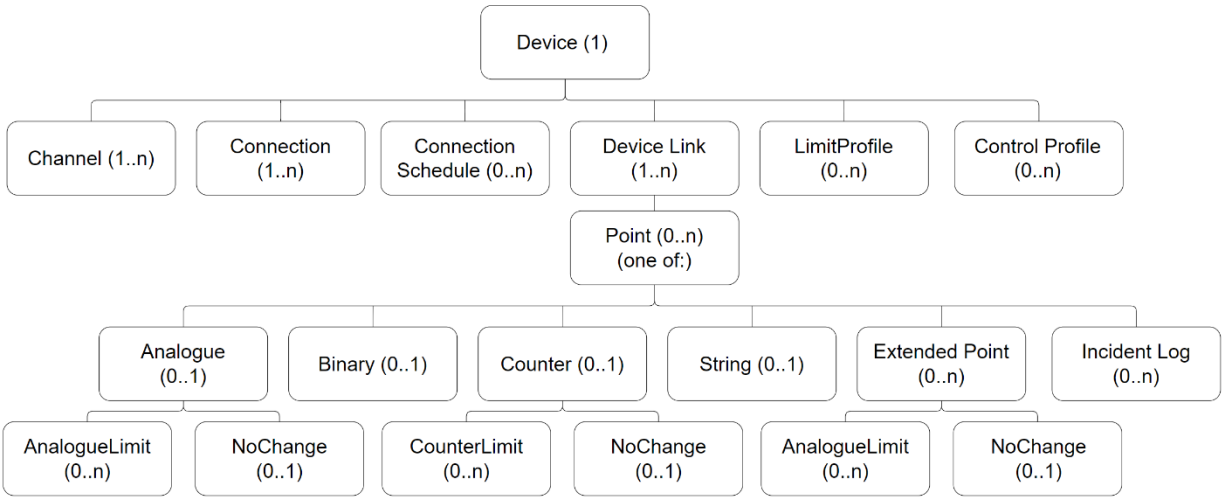


Figure 3: Field device profile object types

All points belong to an FD Link, which has parameters relevant to a group of points. Typically, one FD Link relates to physical points on the device itself, and others may be remotely read I/O, such as a connected PLC. One FD Link refers to calculation points, if supported.

Extended points are point derivations such as run time and averages.

In the Device Profile the parentTypes field refers to the type names of parent objects. It is an array, as more than one parent object type may be permitted.

The minObjects and maxObjects fields declare the number of objects for any one parent, and totalObjects declares the maximum number which can be defined for the FD as a whole. The Incident Log object type can be added to any point type.

Figure 3 also shows which objects are required or optional within a full configuration. An object is optional if the figure shows "0.." and required if the figure shows "1..". So, for instance, a "Connection" is required for a "Device", whereas an "Analogue Limit" is optional for an "Analogue" point.

Where a "Point" is defined then a child object for that "Point" type must be included. So for example, if a "Point" of type analogue input is defined, then there must be an "Analogue" child object to match that point. The reason cardinality is shown as "0.." for "Analogue" on the figure is that it is possible for there to be 0 "Analogue" children if no analogue "Point"s are specified.

2.1.3 Configuration

Configuration can be uploaded from or downloaded to an FD using the topics discussed below. The format of configuration information is detailed in the following sections. Errors in configuration are dealt with as detailed in Section 2.1.3.3, Configuration errors.

2.1.3.1 Upload topic

[UUID]/UP/configuration

The complete configuration is sent from FD to SA on start-up, and when configuration changes are made locally to the FD. The version number is controlled by the FD and is increased automatically. If the SA receives a configuration version number in the status message which does not match its record of the version, then it requests the configuration using the Get Field Device Info message.

2.1.3.2 Download topic

[UUID]/DOWN/configuration

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

If the FD is configurable by the SA, then the FD Profile can be used to generate the FD Configuration data structure. See canConfigDown in the FD Profile which state whether download is possible. Configuration is sent from SA to FD when configuration changes are made by the user of the SA system, or if configRequired is in the status message.

On receipt and acceptance of the configuration, the FD updates the version number for the Configuration and resend the status message containing the number.

The SA may send any part or parts of the configuration payload via the Broker, applied incrementally to the FD.

The expected version number of the configuration is always in the configuration message.

2.1.3.3 Configuration errors

If the FD finds errors in the Configuration message, it discards the message entirely and send a new status message with the unchanged configuration version and a non-empty message within the configState. The content of the message is arbitrary and could range from ”Error” to a description of the specific error. On a successful configuration, the status message is resent with updated version and empty or absent configState.

2.1.3.4 Configuration payload

Refer to the code file **Field Device Configuration.json** for an example Configuration message.

The message starts with two properties:

- The version.
- A Boolean indicating whether this is a full or a partial configuration.

A partial configuration is incremental to the existing configuration of the FD or the SA, while a full configuration implies that the previous configuration for this FD would be erased.

```
{
  "cVer": 123,
  "full": true,
  "source": "62d18075-e5e2-49c2-89e2-abd1cc8025af",
```

The field cVer is the configuration version number.

- cVer is mandatory.
- full, when omitted, defaults to false.

The device only receives and acts on a version which is one greater than the version last received in the status message. If successful, the new version is published by the FD.

To delete items (e.g. a point) in a Field Device a full download must be performed with the deleted items no longer included in the configuration payload. Items cannot be deleted during an incremental download.

The field source is optional and sent by the SA to identify itself as the source of configuration. A GUID is used in the example, but any string could be used by the SA. It can be used by multiple SAs to determine the source of the latest configuration in the status message.

Following the properties, lists of objects are defined, as given by the Device Profile. The first is the FD. While it is an array for consistency with other object types, there is only ever one device entry:

```
"device": [
    {
      "nid": "A1234567890",
      "name": "West Derby PRV",
      "loc": "16 Bridge St, Elton",
      "ownr": "STW",
```


Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

```
      "asst": "ASST123456",
      "serl": "2005-12345Abc",
      "latitude": 12.4356543,
      "longitude": 76.12324,
      "scan": true
    }
  ],
```

Table 1 describes the elements within the object: **device**.

Table 1: Device elements

Information	Type	JSON Name	Mandatory	Description
Node Id	String	nid	Yes	User set name of the device in its application, unique for this user's set of devices. Commonly a numeric code and used to facilitate device change.
Name	String	name	Yes	The name of the device
Location	String	loc	No	Location of the device, such as an address
Owner	String	ownr	No	The owner organisation of the device
Asset	String	asst	No	The asset code of the device
Serial	String	serl	No	The manufacturer's serial number of the device
Latitude	Number	latitude	No	WGS84 latitude. (If Location is True and this is absent then the latitude is zero).
Longitude	Number	longitude	No	WGS84 longitude. (If Location is True and this is absent then the longitude is zero).
On Scan	Boolean	scan	Yes	True to set the device active and able to return data

The nid, name and scan fields are mandatory.

The nid field identifies the device to aid device replacement. While all physical FDs have unique a UUID, when replacing a device, the nid is the same.

The scan field indicates whether the device is active and returns data.

The object types which follow are linked to parent objects through the parent object type name and object index, and if necessary, a type field which, with the parent field, forms a unique reference to the parent.

For example, a point object needs a deviceLink numeric field to refer to its parent deviceLink, and an analogue object needs a point numeric field and a type field to refer to its parent point.

The format of the sections of an FD Configuration message is fully defined by the FD Profile.

2.1.3.5 Channel and connection information

The device defines channels which are physical communications paths from the device to the Broker. The Connection Information entry defines the details of the communications that can occur over these channels. There may be multiple connection entries over a single channel. Each connection defines a route from the device to a Broker over a single channel. The maximum number of connections is defined in the FD Profile. An FD uses its configured connection records as a prioritized list to establish a connection to the Broker.

Table 2 describes the elements within the object: **channel**.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

Table 2: Channel elements

Information	Type	JSON Name	Mandatory	Description
Channel Number	Integer	num	Yes	Storage Index
Channel Name	String	name	Yes	The name of this channel
Connection String	String	conn	No	Optional text required by the device for this channel number. May contain device or hardware specific settings.
DNS Information	String	dns	No	DNS names in a comma-separated string

The numbers and names of pre-defined channels are defined in the Capability section of the FD Profile.

The following fields are defined for a connection entry:

Table 3 describes the elements within the object: **connection**.

Table 3: Connection elements

Information	Type	JSON Name	Mandatory	Description
Connection Number	Integer	num	Yes	Storage Index The number of the connection in the FD.
Connection Name	String	name	Yes	The name of this connection
Channel Number	Integer	channel	Yes	The channel to use
Network Protocol	Integer	prot	Yes	Protocol for network connection The following network protocols are defined: 0 – IPv4 (TCP) 1 – IPv6 (TCP)
Network Information	String	net	Yes	IP address and port number to connect to the Broker
Retries	Integer	retries	No	Number of retries
Backoff	Integer	backoff	No	Retry interval in milliseconds

Channel Number

The channel number defines which communications channel on the FD the connection is associated with. An FD communications channel may be physical (for example, Ethernet or serial) or virtual (for example, a modem on a serial port supporting multiplexed GSM and GPRS connections).

Network Information

The specific network information depends on the network protocol and is used by the FD when it needs to contact the Broker, including when it asserts a “request to connect”. The following connection information is used for each network protocol:

- 0 – IPv4 address or domain name, optionally including port number. When omitted the port number defaults to 1883. The following examples illustrate the valid formats:
- 192.168.0.1
- 192.168.0.1:8883
- ms.your-company.co.uk
- ms.your-company.co.uk:8883
- 1 – IPv6 address or domain name, optionally including port number. When omitted the port number defaults to 1883. The following examples illustrate the valid formats:

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

- ABCD:EF01:2345:6789:ABCD:EF01:2345:6789
- [ABCD:EF01:2345:6789:ABCD:EF01:2345:6789]:8883
- ms.your-company.co.uk
- ms.your-company.co.uk:8883

Retries

If the connection attempt to the Broker fails, then it can be reattempted a number of times. This element specifies how many times the connection attempt can be retried. The number can be between 0 retries and a maximum number of retries as specified in the FD Profile.

Should all these attempts fail then the FD abandons the attempt to connect until such time as another connection is requested.

Backoff

If a connection attempt fails and retries are permitted, then the FD waits a number of milliseconds after the connection attempt before retrying. This element specifies how many milliseconds to wait between each connection attempt. Backoff may vary between 0 and a maximum number specified in the FD Profile. When Backoff is 0 then no delay is inserted between the failed attempt and the next attempt to connect.

2.1.4 Connection scheduling

The FD sends information to the SA via the Broker. It also receives messages from the SA via subscriptions to messages on that same Broker. Two events cause the FD to initiate a connection to the Broker:

- An event can occur within the FD.
  For example, the state of a point changing that is required to be published immediately.
- A scheduled connection occurs.

The scheduled connection entry defines when these scheduled connections should occur. The capability section of the Field Device Profile specifies the number of available connection schedules. The fields are as follows:

Table 4 describes the elements within the object: **connectionSchedule**.

Table 4: Information used in defining scheduled connections

Information	Type	JSON Name	Mandatory	Description
Connection Schedule Number	Integer	num	Yes	Storage Index
Start Time	Integer	start	No	The start time for the scheduled connection, as an offset in milliseconds from the start of the specified boundary).
Repeat Interval	Integer	repeat	No	The number of milliseconds between the start of each connection attempt.
Repeat Count	Integer	count	No	The number of repeats made after the initial connection, during the startBoundary interval
Start Time Boundary	Integer	startBoundary	No	0 = Day (default), 1 = Week, 2 = Month.

Start Time

The start time indicates the first time that an FD must attempt to contact the Broker within a week or month.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

For more information, see Section 2.1.5, Time formats.

Repeat Interval

The repeat interval indicates the period in milliseconds at which the FD must re-attempt to contact the Broker, starting at the Start Time. A repeat interval of zero indicates that the scheduled connection is once per week or month. Scheduled connections repeat at the same time each week. For example, a connection at the start of Tuesday and Friday is achieved using a boundary of a week, a start time of 24 hours, and an interval of three days.

Repeat Count

The number of repeat connections that are made during the startBoundary interval after the initial connection has been made. For example, if the repeat count is two, then there would be three connection attempts (one initial connection plus two repeats). These would be spaced by the repeat interval.

Start Boundary

If 0 (default) the start boundary is a day (from midnight UTC). If 1 the start boundary is a week (from midnight UTC Monday), if 2 the start boundary is a calendar month (from midnight UTC 1st day).

2.1.5 Time formats

Table 5 describes the time formats used within the protocol.

Table 5: Time formats

Time format	Definition
Absolute time	All times are specified in UTC. Times are specified numerically, in milliseconds from 1 Jan 1970 00:00:00 in UTC, as a JSON integer. For example: 1491550944123 (https://en.wikipedia.org/wiki/Unix_time)
Relative time	All durations are specified in milliseconds as a number. For example: Connection backoff time and binary state change persistence.
Weeks	The week is defined to start on Monday at 00:00:00 (that is, midnight between Sunday and Monday in UTC).

2.2 Events

The Events and Actions sections introduce concepts used in Section 2.4, Points.

During the life of an FD events occur that the SA needs to be informed about. Important events need to be sent to the SA immediately. If the event is not important it can be stored, to be sent at a later time, allowing more efficient communications.

The FD maintains an event queue which it uses as the source of information to be delivered over the protocol. The implementation of such an event queue is specific to the FD, apart from the rules relating to events specified in this document.

2.3 Actions

It is possible to associate actions with certain changes in the FD. The changes may be things such as points changing states or point quality flags changing value. Table 6 shows the possible actions.

Table 6: Actions

Action	Description
0	Do nothing
1	Create an event to be stored for a point.
2	Create an event to be stored for a point and trigger a connection request. This can be called an alarm.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

During the life of an FD certain changes in the FD can occur. Where these are important, they may cause an event to be generated. Events allow the FD to inform the SA that the change has happened. The selection changes and whether events should be raised are controlled by the configuration of the device.

For example, if the configured action for the previous state is **2** and the configured action for the new state is **1**, then the previous state action (**2**) executes. It generates an instance of an event and requests a connection.

When the point is on scan and any of the point quality flags change, the action for the flag must be performed to report the new point quality flag. This is subject to the highest permitted action. This does not apply until the Restart point quality flag is cleared, and the state of the point is determined.

For more information, see the Configuration properties whenXXX in Table 8.

In the following situations the highest action of all the states, and all of the point quality flags, must be performed regardless of the highest permitted action. This is needed to report the current state of the point.

- At device start up (and following a restart).
- When the point is put on scan.
- Following a configuration change to any of the point state actions or any of the point quality flag actions.

The action is performed once the point has completed the next or initial scan of its I/O without applying persistence. That is, when the Restart point quality flag is cleared.

When action **2** is taken and a connection request is triggered, the exact action performed by the FD depends on the capabilities, configuration, and state of the FD as follows:

- If the FD can make a connection to the MQTT Broker, and a connection is not already established, then it asserts a “request to connect”. This indicates to the FD that it must make a connection to the MQTT Broker to report the event.
  The “request to connect” is used in the algorithm which the FD uses to decide when it needs to make a connection to the MQTT Broker, along with any other factors, such as retry delays; and blacklists.
- If the FD is already connected to the MQTT Broker, then the event data is immediately published.
- Otherwise, no further action is taken, and the event data is not reported until the FD next connects to the MQTT Broker.

2.4 Points

All points are referenced to a Device Link object. The configuration structure for this is as follows:

Table 7 describes the elements within the object **deviceLink**.

Table 7: Configuration structure for a device link

Information	Type	JSON Name	Mandatory	Description
Device Link Number	Integer	num	Yes	Storage Index
Name	String	name	Yes	Name of this Device Link
Device Link Connection String	String	conn	No	Optional device-specific string

The Device Link Connection String may contain device or hardware specific settings.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

The numbers and names of pre-defined Device Links are defined in the FD Profile in the Capability section.

There is a name for internal points, those without a physical data source/target which can have values set or overridden. The name is “Internal”, and support is optional within the Device Profile.

There is a name for calculated points, “Calc”, and support is optional within the Device Profile. There are point fields which relate to “Calc” points that are calculated by the FD using the Polish Notation (PN) specification, see Section 2.7, Calculated points.

Table 8 describes the elements within the object: **point**.

Table 8: Configuration structure for a point

Information	Type	JSON Name	Mandatory	Description
Device Link	Integer	parent	Yes	A reference to the Device Link for this point.
Point Number	Integer	num	Yes	A unique index of the point within this point type.
Type	String (enum)	type	Yes	The point type code.
Name	String	name	No	The name is optional and is a string which can be used by the device to report data values or to display locally. Furthermore, the name string may contain a hierarchy of fields with a separator which can imply a structure.
Point Class	String	className	No	The class is optional and is a string which can be used by the SA to denote equipment or asset ontology.
Scan Flag	Boolean	scan	Yes	On-scan is set to True if the point is active (or needs to be made active).
Can Override	Boolean	canOverride	No	Whether local override is permitted.
Scan Rate	Integer	scanRate	No	Base periodic read rate of the source data/hardware. In milliseconds. Also used if present to specify the calculation rate for Calculation points. If not specified the device may use its own rate, but is the faster of the logRate or any attached incidentLog rates.
Scan Offset	Integer	scanOffset	No	Offset time from the start of the week that the scanning is synchronised from (Monday, 00:00:00). Measured in milliseconds. If 0, then log is offset from the last midnight. Also used if present to specify the calculation offset for Calculation points.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

Information	Type	JSON Name	Mandatory	Description
Log Rate	Integer	logRate	No	Base periodic logging rate (absent or 0=no logging). In milliseconds.
Log Offset	Integer	logOffset	No	Offset time from the start of the week that the log is synchronised from (Monday, 00:00:00). Measured in milliseconds. If 0, then log is offset from the last midnight.
When Connected	Integer	whenConnected	No	When the device has successfully contacted the Broker, the value is sent to the SA if this action is 1. (If action 2 is selected then it will not contact the SA and will behave as for action 1).
When Online	Integer	whenOnline	No	The action to be taken when this flag changes (0=do nothing, 1=log event with reason, 2=log event and trigger a connection to the SA).
When Restarted	Integer	whenRestart	No	The action to be taken when this flag changes (0=do nothing, 1=log event with reason, 2=log event and trigger a connection to the SA).
When Commu-nications Lost	Integer	whenCommsLost	No	The action to be taken when this flag changes (0=do nothing, 1=log event with reason, 2=log event and trigger a connection to the SA).
When Remotely Forced	Integer	whenRemForce	No	The action to be taken when this flag changes (0=do nothing, 1=log event with reason, 2=log event and trigger a connection to the SA).
When Locally Forced	Integer	whenLocForce	No	The action to be taken when this flag changes (0=do nothing, 1=log event with reason, 2=log event and trigger a connection to the SA).
When Chatter	Integer	whenChatter	No	The action to be taken when this flag changes (0=do nothing, 1=log event with reason, 2=log event and trigger a connection to the SA).
When Overrange	Integer	whenOverrange	No	The action to be taken when this flag changes (0=do nothing, 1=log event with reason, 2=log event and trigger a connection to the SA).

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

Information	Type	JSON Name	Mandatory	Description
When Discontinuity	Integer	whenDiscontinuity	No	The action to be taken when this flag changes (0=do nothing, 1=log event with reason, 2=log event and trigger a connection to the SA).
When Reference Error	Integer	whenRefError	No	The action to be taken when this flag changes (0=do nothing, 1=log event with reason, 2=log event and trigger a connection to the SA).
When UnableTo Read	Integer	whenUnableToRead	No	The action to be taken when this flag changes (0=do nothing, 1=log event with reason, 2=log event and trigger a connection to the SA).
When Changed	Integer	whenChange	No	If the point has changed value. The action may be 0=do nothing (default), 1=log event, 2=log event and trigger a connection to the SA. Behaviour varies slightly with type: Analogue and counter values change by a significant change deadband. Strings can include any change. Digital points have their own change requirement per state.

An FD can support various types of points. See Table 9 for further information.

Table 9: Supported point types and identifiers used within the protocol

Point type	Object name	Identifier
Analogue input	analogue	ai
Analogue output	analogue	ao
Binary input	binary	bi
Binary output	binary	bo
Counter	counter	c
String	string	s
Extended point	extPoint	x

The point type identifier is combined with a number to uniquely identify every point on an FD and is an efficient way to refer to points. The number should be a whole number (0 or above). The unique identifier is known as the point identifier. For example, **ai0** is the point identifier for analogue input 0 and **s4** is the point identifier for string point 4.

Points can optionally have a point name. If defined this is a string which identifies the point and is unique on the FD. When specifying a point within the protocol either the point identifier or point name can be used.

The only way to set a point on or off scan is to use the configuration message.

The class is optional and is a string which can be used by the SA to denote equipment or asset ontology.

Point values for analogues, binaries, counters, and extended points are normally integers or reals. Binary inputs and outputs are numeric values such as 0, 1, 2, and so on. String points are represented by normal

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

JavaScript strings. Strings encoded as non-text MIME types are represented by a value encoded as a BASE64 string.

Point values are updated at various times. For native binary and counter points these may be updated immediately their value changes. However, in some cases point values may only be updated at specific times. For example, an analogue input point may only have its value taken occasionally, or a point may be read over a Modbus interface which is only scanned at a certain rate.

The rate at which point values are taken determines the values used in point state evaluation and calculation of extended point values. To ensure users are aware of when point values are available, each point can have a default update period defined. Without this definition a point value would be updated when a log value is requested or when a read is requested via the protocol. A default update period of 0ms would mean that a point is updated immediately it changes. An optional separately specified scan rate and offset are available to specify the hardware read rate, useful for extended points such as maximum or minimum.

Extended points are time-based derivations from other points such as Minimum or Integrator.

2.4.1 Point quality flags

Each point has accompanying flags providing quality information about the point. The flags accompany the point value as a bitmap and give extra information about the value, either the quality of the input or the reason the value was logged. The point quality flags are defined in Table 10.

Table 10: Point quality flags

Name	Bit Number	Description
Offline	0	Indicates that the point is not active and available. If offline is set, then the point is off scan. While online the point behaves normally. Offline is set if the point is not yet active, disabled, or off scan. For example, a point may not be active or be disabled because it is faulty or powered down.
Restart	1	Indicates that a point is restarting. This is set when the device is restarted. It may also be set when the point is reconfigured. For an input point, restart indicates that the point has not yet provided a first data value. Once that value has been provided, restart is reset. For an output value, restart indicates that the point cannot accept a command. Restart is reset once the output point can accept commands.
Comms Lost	2	Indicates that there is a communications failure in the path used to access data from the point by the FD. Data from the point cannot be relied on in this case and the last known value is assumed to be stale. When communications with the point is returned Comms Lost is reset.
Remote Forced	3	Indicates that read from a device connected to the FD is being overridden by that device or a device downstream from that device.
Local Forced	4	Indicates that the value of this point is being overridden by the FD. This may be due to human intervention, a diagnostic mode of the device, the point being overridden by the protocol, or some other reason.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

Name	Bit Num-ber	Description
Chatter	5	This flag applies to Binary Inputs only. This flag indicates that a chatter filter is being applied by the FD to the binary input. The chatter filter allows the FD to reduce the number of events generated by the device when the input is changing quickly. The exact definition of what changing quickly means is device dependant and not something that can be changed by the protocol. When the chatter filter is being applied the points value does not necessarily represent the true value of the point as the value may be forced by the filter.
Range Error	6	This flag applies to analogue inputs and outputs only. It indicates that the value of the analogue point has exceeded the valid range for that point. If the full-scale limits of the convertor used to produce the analogue value are not exceeded, then the value of the point reported is the actual value, not limited to the valid range of the point. If the full-scale limits of the convertor are exceeded, then the value reported is the value of the relevant full-scale limit.
Discontinuity	7	This flag applies to Counters only. It indicates that the value being reported with the flag cannot be used with the previous value to give a count difference. For example, this applies if a counter value was reset.
Reference Error	8	This flag applies to analogue inputs and outputs only. It indicates a problem with the measurement process for the point which results in the point value being inaccurate or incorrect.
Unable to Read	9	This flag applies to analogue inputs and outputs only. It indicates a problem with the measurement process for the point which results in the point value being unavailable.

The point quality flags can be configured to generate events. Refer to the fields whenXXXX in Table 8, for more information. If a point quality flag generates an event, then the reason code will be set to “State Change” assuming there is no other valid reason behind the event. See Section 4.1 for more details of the reason codes.

The values used when quality flags are set are usually the last known value, or zero (or equivalent) if the value was never known.

2.4.2 Binary points

The term binary is used in this document to describe the point, and as the term “bi” and “bo” for binary input and binary output.

Table 11 describes the elements within the object: **binary**.

Table 11: Configuration structure for a binary point

Information	Type	JSON Name	Mandatory	Description
Parent Point	integer	parent	Yes	Binary features belong to binary parent points only.
Parent Point Type	string	parentType	Yes	“bi” or “bo”
Bit Count	integer	bits	No	Bit count, defaults to 1. Values are 1,2,3 only.
State 0 Action	integer	state0Action	No	0=do nothing, 1=log data, 2=log data and connect to SA.

Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

Information	Type	JSON Name	Mandatory	Description
State 0 Persistence	integer	state0Pers	No	Persistence – number of milliseconds the point must remain in this state before the action is taken. (Optional).
State 0 Name	string	state0Name	No	Name - name of state, may be used locally. (Optional).
State 1 Action	integer	state1Action	No	0=do nothing, 1=log data, 2=log data and connect to SA.
State 1 Persistence	integer	state1Pers	No	Persistence – number of milliseconds the point must remain in this state before the action is taken. (Optional).
State 1 Name	string	state1Name	No	Name - name of state, may be used locally. (Optional).
State 2 Action	integer	state2Action	No	2=do nothing, 1=log data, 2=log data and connect to SA.
State 2 Persistence	integer	state2Pers	No	Persistence – number of milliseconds the point must remain in this state before the action is taken. (Optional).
State 2 Name	string	state2Name	No	Name - name of state, may be used locally. (Optional).
State 3 Action	integer	state3Action	No	0=do nothing, 1=log data, 2=log data and connect to SA.
State 3 Persistence	integer	state3Pers	No	Persistence – number of milliseconds the point must remain in this state before the action is taken. (Optional).
State 3 Name	string	state3Name	No	Name - name of state, may be used locally. (Optional).
State 4 Action	integer	state4Action	No	0=do nothing, 1=log data, 2=log data and connect to SA.
State 4 Persistence	integer	state4Pers	No	Persistence – number of milliseconds the point must remain in this state before the action is taken. (Optional).
State 4 Name	string	state4Name	No	Name - name of state, may be used locally. (Optional).
State 5 Action	integer	state5Action	No	0=do nothing, 1=log data, 2=log data and connect to SA.
State 5 Persistence	integer	state5Pers	No	Persistence – number of milliseconds the point must remain in this state before the action is taken. (Optional).
State 5 Name	string	state5Name	No	Name - name of state, may be used locally. (Optional).
State 6 Action	integer	state6Action	No	0=do nothing, 1=log data, 2=log data and connect to SA.
State 6 Persistence	integer	state6Pers	No	Persistence – number of milliseconds the point must remain in this state before the action is taken. (Optional).
State 6 Name	string	state6Name	No	Name - name of state, may be used locally. (Optional).

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

Information	Type	JSON Name	Mandatory	Description
State 7 Action	integer	state7Action	No	0=do nothing, 1=log data, 2=log data and connect to SA.
State 7 Persistence	integer	state7Pers	No	Persistence – number of milliseconds the point must remain in this state before the action is taken. (Optional).
State 7 Name	string	state7Name	No	Name - name of state, may be used locally. (Optional).
Control Pulse Time	integer	controlPulseTime	No	Control Pulse time (milliseconds) applies only to 1 bit binary output points.
Control Pulse Polarity	boolean	controlPulsePolarity	No	Control Pulse polarity applies only to 1 bit binary output points. If True, then the pulse is from 1 to 0 and back to 1. If False, then the pulse is from 0 to 1 and back to 0.

2.4.2.1 Multi-bit binary inputs and outputs

It is possible for binary inputs and outputs to be grouped together to provide a two-bit binary or three-bit binary. The “bits” parameter specifies the number of bits used from 1 to 3. If not specified, then 1 is implied. The point number refers to the least significant bit, and consecutive bits are more significant.

For example, a two-bit binary input using the number 8 will include number 9. When the data is read for this point, both bits are included, and the higher bits cannot be read. In this example bit number 9 is invalid.

For example, a three-bit binary input using the number 10 has the third, most significant, bit number 12.

2.4.2.2 Binary state persistence

The state persistence defines the number of milliseconds the point must remain in state # before the state # action is performed.

When multiple changes occur in one scan of I/O for a point (state, point quality flag or other changes) only a single action (the highest action of all the changes that occurred) is performed by the FD subject to the highest permitted action. This ensures only one action occurs for a single I/O scan.

When a binary point changes, an FD immediately updates its database with the new ***value*** of the point. When any configured persistence expires, and there are no further changes of the point, the FD reports the point change of state to the SA with its current ***value***. This implementation method means that any entity, such as the SA or local application programs, that accesses the FD’s database before the persistence has expired, sees the new ***value*** of the point – even though it may not have persisted for the configured time.

Figure 4 presents the diagrammatic view.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

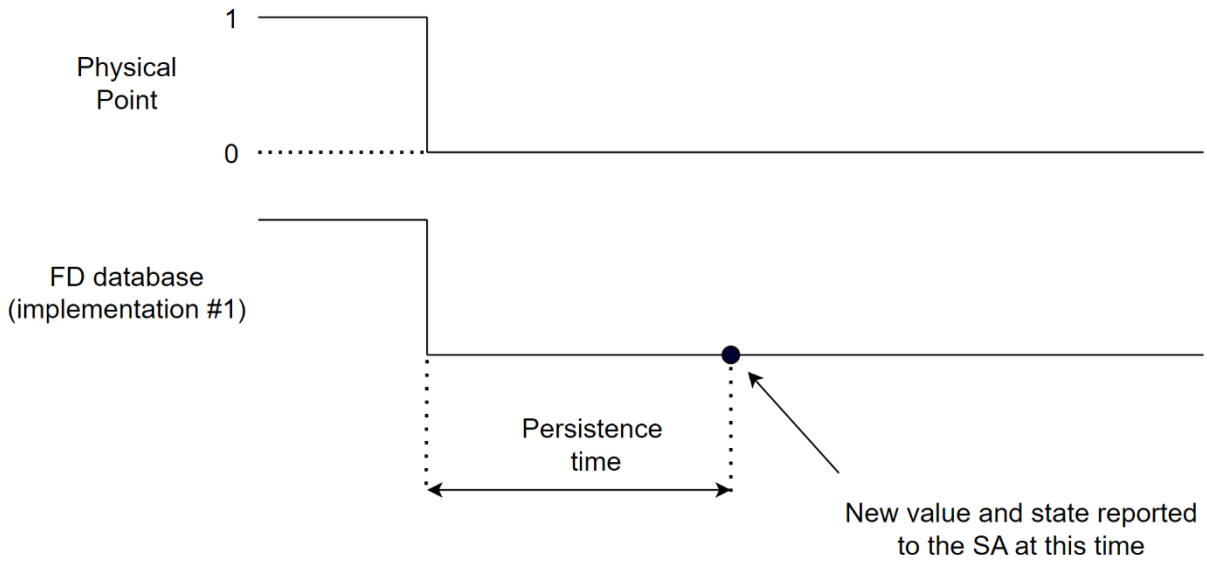


Figure 4: Supported implementation of persistence for binary points

2.4.2.3 Pulsed controls

Only a 1 bit binary output point can support pulsed controls. The default pulse time is specified and is non-zero to indicate that pulsed controls are possible. The control pulse polarity field can be used to reverse the output pulse value so that it is normally on.

2.4.3 Analogue points

Table 12 describes the elements within the object: **analogue**.

Table 12: Configuration structure for an analogue point

Information	Type	JSON Name	Mandatory	Description
Parent Point	integer	parent	Yes	Analogue features belong to analogue parent points only.
Parent Point Type	string	parentType	Yes	“ai” or “ao”
Raw Scale 1	Integer	rawsc1	No	Raw Scale 1 is the first raw value of the point (e.g. lower or zero value for the instrument).
Raw Scale 2	Integer	rawsc2	No	Raw scale 2 is the second raw value for the scaling (e.g. higher or full scale value of the instrument).
Eng Scale 1	number	engsc1	No	Engineering Scale 1; this is the first engineering value that corresponds to the first raw value (e.g. smallest scaled output that can be reported.
Eng Scale 2	number	engsc2	No	Engineering scale 2; this is the second engineering value that corresponds to the second raw value (e.g. the highest scaled value that can be recorded).

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

Information	Type	JSON Name	Mandatory	Description
Minimum	number	min	No	The measurement range's minimum value, in engineering units. If a value is below the minimum, then it is under-range. The OVER_RANGE flag must be set when reporting values that are under-range. Under-range values should not be used when determining significant data (dead-band) changes.
Maximum	number	max	No	The measurement range's maximum value, in engineering units. If a value is above the maximum, then it is over-range. The OVER_RANGE flag must be set when reporting values that are over-range. Over-range values should not be used when determining significant data (dead-band) changes.
Units	string	units	No	The units are a free format ASCII string that can be used when displaying the point value through local access displays etc. The FD should not interpret any meaning from this string and may choose to ignore it completely.
Significant Change Value	number	change	No	The value difference (a positive value) used to determine when the point has changed by a significant amount.

2.4.3.1 Analogue point scaling

Analogue inputs and outputs and counters may be scaled. If no scale is defined, then a 1:1 scale with no offset is assumed.

Scaling is optional, and if these parameters are not used then the device reports its raw value.

Raw Scale 1 and Engineering Scale 1 are associated, as are Raw Scale 2 and Engineering Scale 2.

Figure 5 highlights the relationship.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

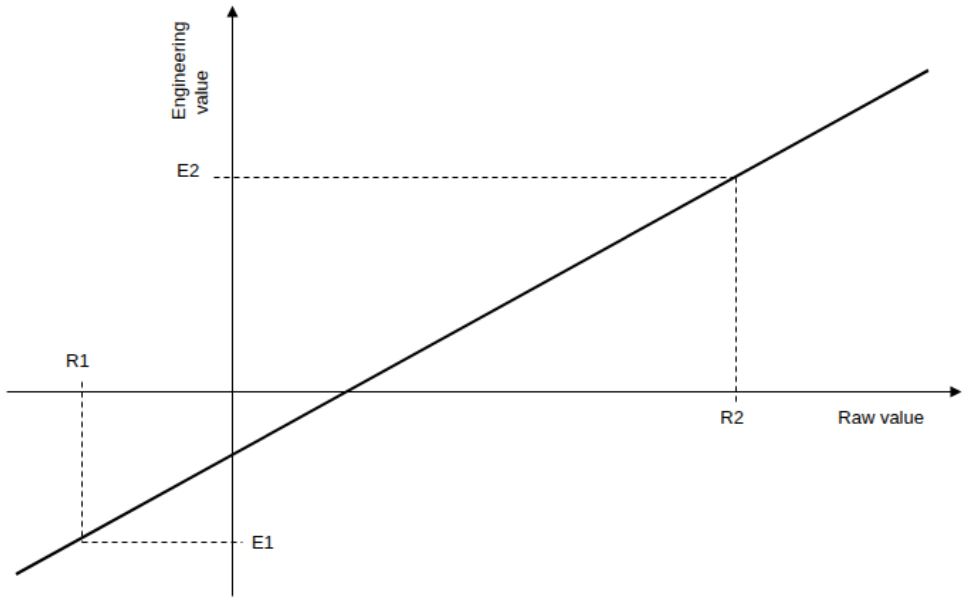


Figure 5: Analogue scaling

The scaling values may be inverted such that Raw Scale 1 is greater than Raw Scale 2, or Engineering Scale 1 is greater than Engineering Scale 2, giving a graph with a negative slope.

The raw and engineering values need not be within the range for the point.

If scaling is not supported for a particular point, for example, a point whose value is fixed by the FD, then the FD rejects any configuration which is not 1:1 scaling.

Changing the scale of a point doesn't affect the range, limit values, and hysteresis defined for the point. For example, the SA can download a configuration containing limit records. Later, it may download a second configuration containing just a change to the scaling, but this does not affect the limits already stored from the earlier configuration. On the next I/O scan the point may change state due to the new engineering value being evaluated against the existing limits.

2.4.3.2 Range

Analogue points may have a valid range defined. See the fields **min** and **max**.

The value reported for an analogue point which is over or under range must be the over-range or under-range value. If a value reaches an instrumentation limit, then the instrumentation limit is reported.

A FD implements proprietary procedures to ensure it does not generate events that report fleeting changes of the Range Error point quality flag.

2.4.3.3 Units

Analogue points may have units defined. See the field **units**.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

2.4.3.4 Significant change

Analogue points support the generation of events when point values differ from previous recorded values by a certain amount. These are known as significant change events.

A significant change event is generated when the absolute value of the difference between the last evented value and the current value of the point exceeds a set limit. The last evented value of the point must be the last value evented for a significant change. If no such value is available then the first recorded value of the point must be used, that is, the value when the Reset point quality flag is cleared. Significant change detection is performed on the scaled value of the point.

The significant change value is the non-negative change of point value that is considered to be significant. The significant change value is specified in engineering units, and the FD stores it in engineering units, not the equivalent raw value, so that it is not affected by any subsequent change to the scaling of the point.

A significant change value of 0 turns off significant change detection for that point.

Significant change detection must not be performed if the point has either the Restart or Range Error point quality flag asserted.

The action to be taken is described in Section 2.4, Points. The **whenChange** property of the point determines the action.

2.4.4 Counter points

Table 13 describes the elements within the object: **counter**.

Table 13: Information used in counter points

Information	Type	JSON Name	Mandatory	Description
Parent Point	integer	parent	Yes	String features belong to String parent points only.
Reset Period	integer	resetPeriod	No	If zero or absent, then is not reset. Interval between resets of this point's value calculation, in milliseconds).
Reset Offset	integer	resetOffset	No	Offset from start of week (Monday) in milliseconds.
Report Period	integer	reportPeriod	No	Interval between reports of this point's value. In milliseconds, starting at reset with last accumulated value.
Scale Value	number	scale	No	Multiplies the number of counts to evaluate the value of this counter. i.e. this is the scaled value of one count.
Units	string	units	No	The units are a free format ASCII string that can be used when displaying the point value through local access displays etc. The FD should not interpret any meaning from this string and may choose to ignore it completely.
Significant Change Value	number	change	No	The value difference (a positive value) used to determine when the point has changed by a significant amount.

2.4.4.1 Units

Counter points may have units defined. See the field **units**.

2.4.4.2 Counter point scaling

Scaling is optional, and if these parameters are not used, then the FD reports its raw value.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

1. **Significant change**

   Counter points support the generation of events when point values differ from previous recorded values by a certain amount. These are known as significant change events.

   A significant change event is generated when the absolute value of the difference between the last evented value and the current value of the point exceeds a set limit. The last evented value of the point must be the last value evented for a significant change. If no such value is available then the first recorded value of the point must be used, that is, the value when the Reset point quality flag is cleared. Significant change detection is performed on the scaled value of the point.

   The significant change value is the non-negative change of point value that is considered to be significant. The significant change value is specified in engineering units, and the FD stores it in engineering units, not the equivalent raw value, so that it is not affected by any subsequent change to the scaling of the point.

   A significant change value of 0 turns off significant change detection for that point.

NOTICE	**Significant change detection must not be performed if the point has either the Restart or Range Error point quality flag asserted.**

   The action to be taken is described in Section 2.4, Points. The **whenChange** property of the point determines the action.

2.4.5 Analogue limits

Analogue points may have multiple states that are separated by limits. The properties of the limit control how and when transitions are made between states. A single point may have multiple limits and therefore multiple states. This section describes analogue limits and how they relate to states.

The information in Table 14 is used to define limits for analogue points within the object: **analogueLimit**.

Table 14: Information used in defining an analogue limit

Information	Type	JSON Name	Mandatory	Description
Parent	Integer	parent	Yes	Analogue point number.
Parent Point Type	String	parentType	Yes	Type of point the limit refers to, “ai”,“ao”,“x”.
Point Limit Index	Integer	num	Yes	Point Limit Index (PLI) for limit.
Name	String	name	No	A human readable name for the limit.
Action	Integer	action	Yes	Action to be taken on entering this state (0=do nothing, 1=log event with reason, 2=log event and trigger a connection to the SA).
Direction	Integer	dirctn	Yes	The direction for entering the state (0=positive going (low-normal-high, 1=negative going (high-normal-low)).
Limit Value	Number	value	Yes	The value of this limit, in engineering units.
Hysteresis	Number	hyst	No	Limit hysteresis, default 0.
Enter Persistence	Integer	enter	No	Limit entry persistence in milliseconds.
Leave Persistence	Integer	leave	No	Limit leave persistence in milliseconds.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

Information	Type	JSON Name	Mandatory	Description
Monday Profile	Integer	mon	No	Limit Profile number for Monday.
Tuesday Profile	Integer	tue	No	Limit Profile number for Tuesday.
Wednesday Profile	Integer	wed	No	Limit Profile number for Wednesday.
Thursday Profile	Integer	thu	No	Limit Profile number for Thursday.
Friday Profile	Integer	fri	No	Limit Profile number for Friday.
Saturday Profile	Integer	sat	No	Limit Profile number for Saturday.
Sunday Profile	Integer	sun	No	Limit Profile number for Sunday.

For example:

```
"analogueLimit": [
  {
    "parent": 1,
    "parentType": "ai",
    "num": 1,
    "name": "High",
    "action": 2,
    "dirctn": 0,
    "value": 123.45,
    "sat": 1,
    "hyst": 1.0,
    "enter": 2000,
    "leave": 2000
  }
],
```

To configure multiple limits on the same point the configuration must contain multiple limit configuration records. An FD supports a change of state mechanism compatible with the diagram shown in Figure 6. Although values or combinations of values for direction, action, hysteresis, enter persistence, and leave persistence are optional, the record always contains every element.

If an element value or combination is not supported by the FD, then the SA must set these values appropriately. If a record contains unsupported values, then the FD must not accept the configuration. When an SA is configuring a sub-set of the limits on a point, it must ensure the limits do not invalidate existing limits. An SA may decide to download all the limits for a point even if some of the limits remain unchanged.

The following rules apply to all enabled limits for an individual analogue point. Disabled limits are ignored for the purposes of these rules and need not be present on upload.

- The limits must be set in ascending order of limit value with Point Limit Index (PLI) otherwise the FD rejects the configuration. When a profile is being used every day of the week there is no static limit value being used. In this case the unused limit value can contain any arbitrary value and must not be included in any validation checks done by the FD.
- Limits with a positive direction must have a higher PLI than limits with a negative direction, otherwise the FD rejects the configuration.
- Hysteresis should not cause an overlap, that is, hysteresis cannot artificially make limits that are not in ascending order with PLI, otherwise the FD rejects the configuration.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

Figure 6 shows how the elements of the record are used to configure an analogue limit.

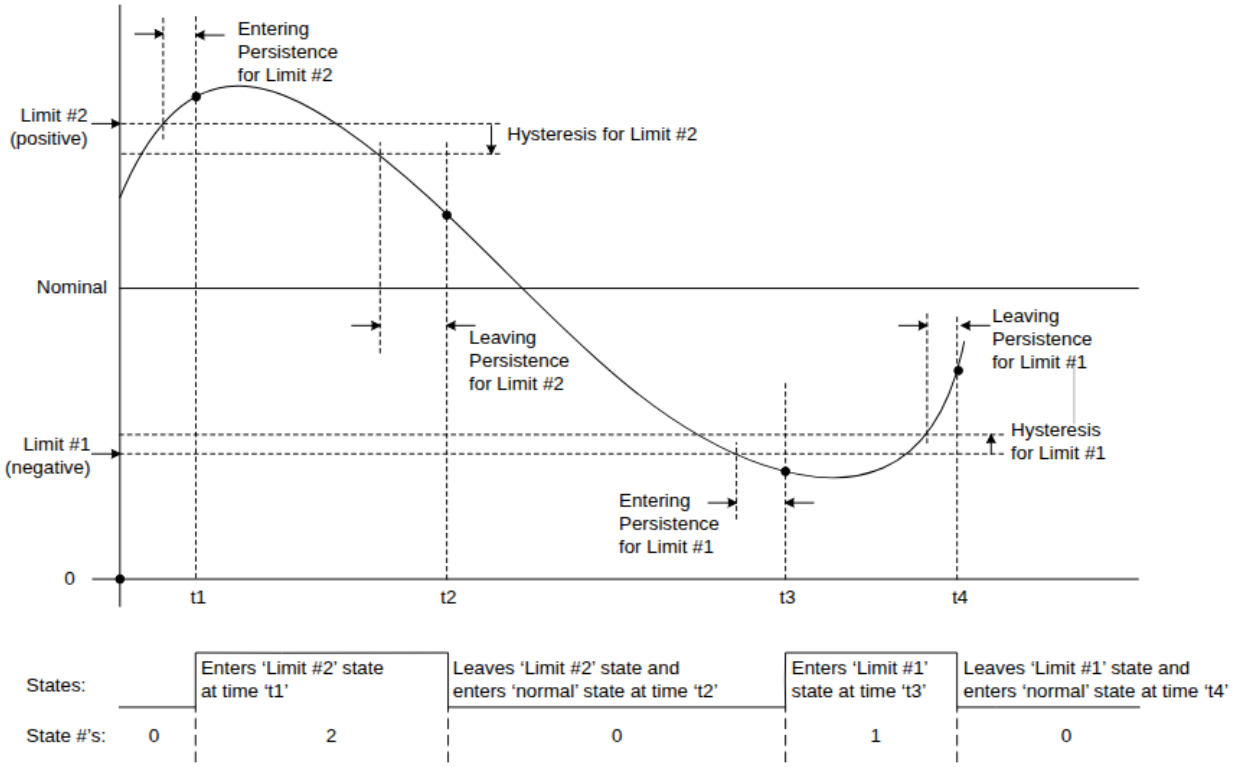


Figure 6: State change mechanism

A state of an analogue point corresponds to the region between two adjacent limits, or the region above the highest limit, or the region below the lowest limit. The region between the lowest positive limit and highest negative limit is referred to as the normal state. If a point has no negative limits, then the normal state is for values below the smallest positive limit. If a point has no positive limits, then the normal state is for values above the largest negative limit. A point without any limits is always in the normal state.

Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

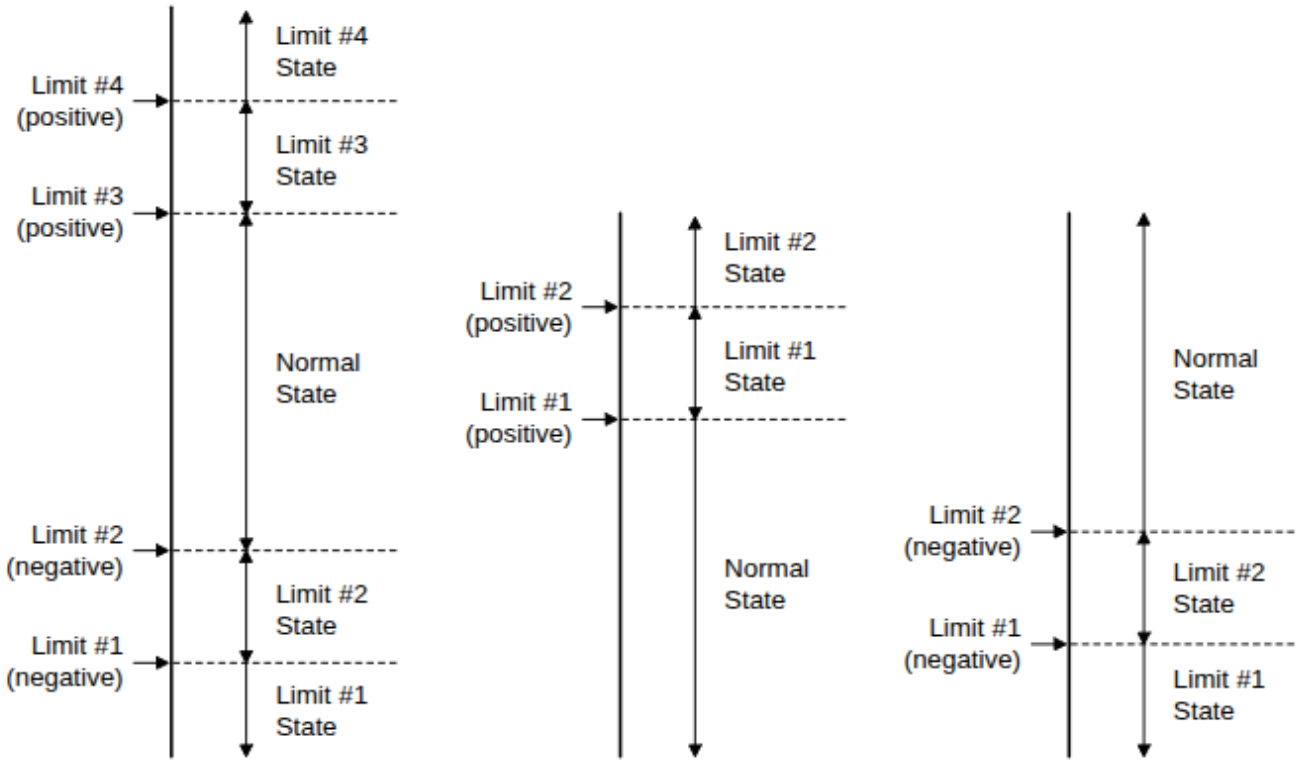


Figure 7: Analogue normal state

For positive-going limits, that is, high limits, the range of values represented by a state includes the value of the lower bound, but not the value of the upper bound.

For negative-going limits, that is, low limits, the range of values represented by a state includes the value of the upper bound, but not the value of the lower bound.

For example, given a point with a negative limit at 20.0 and a positive limit at 80.0:

- The normal state represents values between 20.0 and 80.0 (exclusive).
- The high state represents values greater than or equal to 80.0.
- The low state represents values less than or equal to 20.0.

A point may enter a particular state because of exceeding either of the limits that bound the state.

When an FD supports a separate persistence per limit, then it also supports non-sequential persistence. Non-sequential persistence is where, for example, a series of positive limits have persistence times that get shorter when going positively from one limit to the next. Similarly, non-sequential persistence is where a series of negative limits have persistence times that get shorter when going negatively from one limit to the next. When a limit is exceeded and non-sequential persistence is configured, the action is performed on expiry of the shorter persistence time. Examples of non-sequential persistence are shown in Figure 8 and Figure 9.

NOTE:	**If an FD also supports separate enter state and leave state persistence, then both the enter state persistence and the leave state persistence may be non-sequential. For example, when the point changes from a low state to a high state (or vice versa) and the leave persistence time of the old state and enter persistence time of the new state are different, then the action is performed when the shorter of the two persistence times expires.**

When a point that has persistence configured changes state, the time stamp of the event is the time at which the persistence elapses, not the time at which the limit was initially exceeded.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

Figure 8 illustrates the behaviour as an analogue point rises through four positive limits where the third limit's persistence is not sequential.

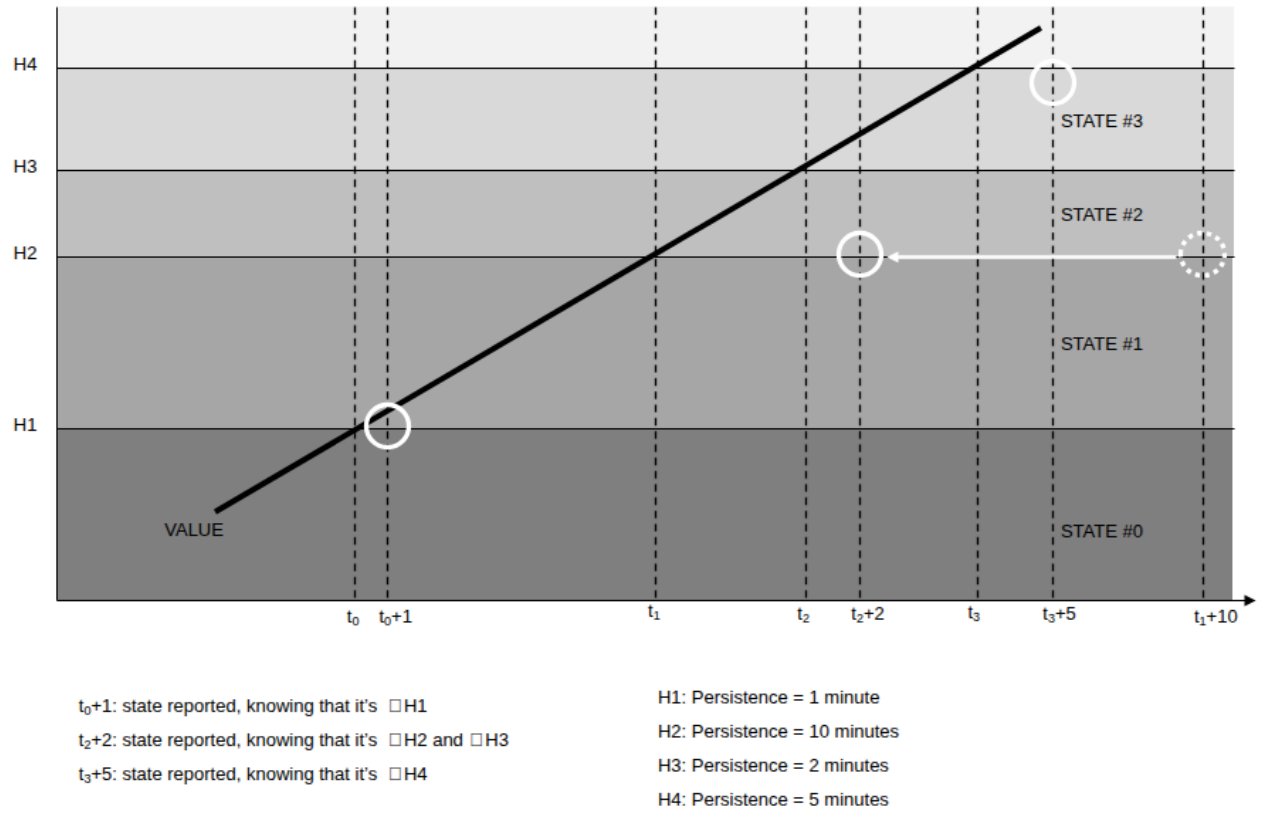


Figure 8: Ramping analogue value with 4 limits

Figure 9 illustrates the behaviour as an analogue point step changes through four positive limits where the third limit's persistence is not sequential.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

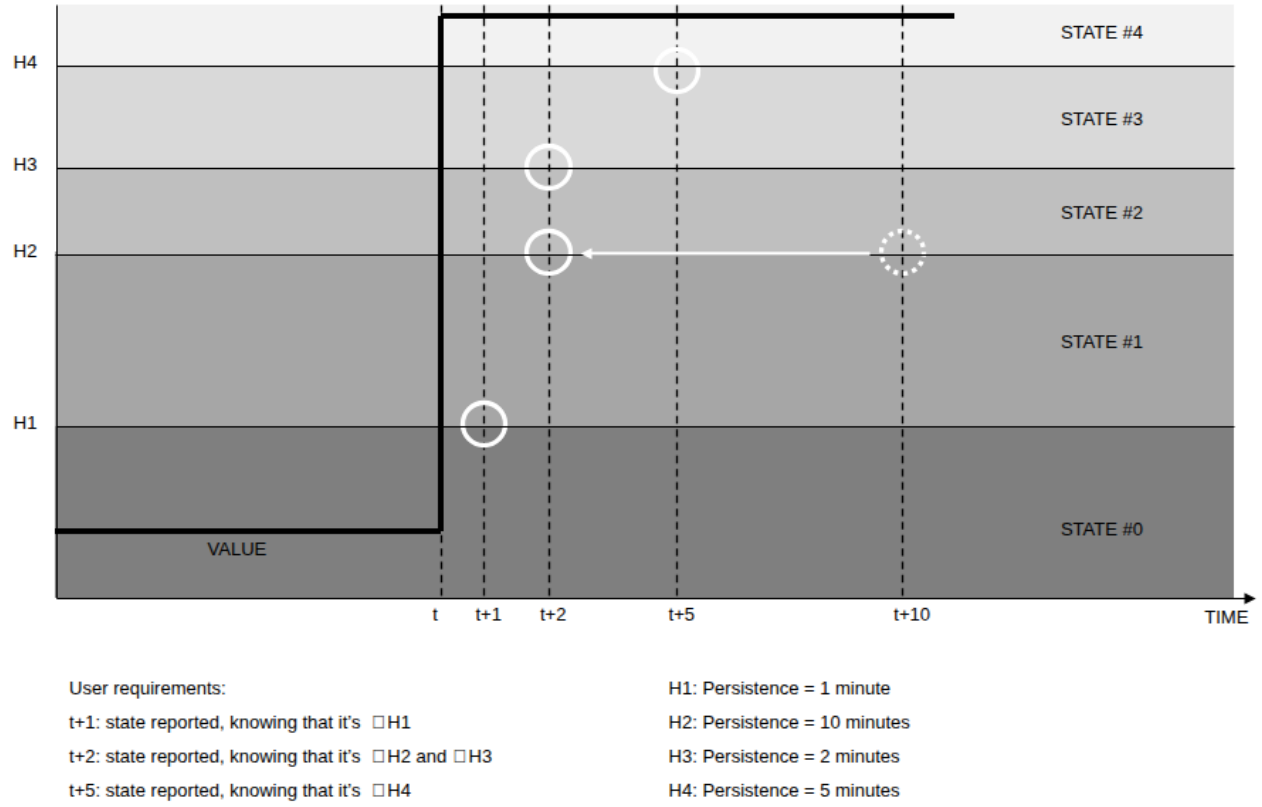


Figure 9: Step-change analogue value with 4 limits

When an analogue point changes state, only the new state of the point is reported. Any intermediate states that may have been exceeded at the time in the change from the old state to the new state are not reported. This ensures that only a single event is generated or logged for each individual state change. For example, if a point has two high limits and changes from the normal state to the high-high state, then one event is generated or logged for entering the high-high state.

When multiple changes occur in one scan of I/O for a point, state, point quality flag or other changes, only the highest action of all the changes that occurred is performed by the FD, subject to the highest permitted action. This is to ensure only one action occurs for a single I/O scan.

If an FD only supports a single hysteresis or single persistence per point, then the SA must download the same hysteresis or persistence value for each limit per point. If the SA downloads different hysteresis or persistence, then the FD rejects the configuration. When the hysteresis or persistence is changed at the SA, it must download all the limits.

2.4.5.1 Point limit index

The Point Limit Index (PLI) defines which of the point's limits the record relates to and must be a valid limit for the FD.

The PLIs used by the FD must start from one and be consecutive. Also, PLI one must correspond to the lowest limit, and the maximum PLI must correspond to the highest limit. PLI zero is reserved to represent the normal state of a point. A maximum number of limits for each analogue point is declared in the SA Profile.

For example, an FD with four limits that have fixed directions may use the following PLIs:

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

- 1 – Low-Low (LL)
- 2 – Low (L)
- 3 – High (H)
- 4 – High-High (HH)

2.4.5.2 Direction

The Direction is the direction in which the point value is moving past the limit value to be entering the corresponding state. Options are:

- 0 for positive going, a high limit.
- 1 for negative going, a low limit.

The direction of each limit in an FD may be fixed, for example, HH, H, L, LL. In this case the SA must download these directions. The FD rejects any configuration with a different direction for a particular limit.

2.4.5.3 Limit value

The limit value defines the threshold for the limit. The limit value is in engineering units, and the FD must store it in engineer units, not the equivalent raw value, so that it is not affected by any subsequent change to the scaling of the point. The FD rejects any configuration that has the same Limit Value on adjacent Point Limit Indices.

When a profile is specified for a day of the week, it overrides the limit value for that day. When a profile is being used every day of the week then the limit value is not used and can contain any arbitrary value. In this case the FD must not perform any validation using the limit value.

2.4.5.4 Action

The Action defines the action to be taken on exceeding the limit in either direction; the action is as defined in Section 2.6, Limit and control profiles.

When a point changes state, the action that is performed by the FD is determined by the highest configured action of the enabled limits that were exceeded as the result of the change:

- If any of these actions are configured as value 2, event and request connection, then the FD must create an event for the new state and assert **request to connect**, so the SA may raise or clear the required alarm(s).
- If none of these actions are configured as value 2, but any of them are configured as value 1, event, then the FD must store an event for the new state.
- Only if all these actions are configured as value 0 (do nothing), the FD performs no action.

When a point is on scan and any of the point quality flags change, the action for the flag must be performed subject to the highest permitted action, to report the new object flags. However, this does not apply until the Restart Point quality flag is cleared, and the state of the point determined.

In the following situations the highest action of all of the enabled limits and all of the point quality flags must be performed regardless of the highest permitted action. This highest action must also include any action configured for significant change of value reporting. This is required to report the current state of the point.

- At device start up and following a restart.
- When the point is put on scan.
- Following a configuration change to the action for any limit or any of the point quality flags.
- When a change in the configuration of the point's limits causes the point to change state.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

The action is performed once the point has completed the next or initial scan of its I/O, that is, when the Restart Point quality flag is cleared, without applying hysteresis or persistence.

2.4.5.5 Hysteresis

The hysteresis is an adjustment made to the limit value to set a further level beyond which the point value must travel to leave the state. The value of hysteresis is always a positive number. The method by which the hysteresis is combined with the limit value to produce the leaving level is dependent on the limit direction.

- For positive going limits the hysteresis is subtracted from the limit value to create the leaving level.
- For negative going limits the hysteresis is added to the limit value to create the leaving level.

2.4.5.6 Enter persistence

The enter persistence defines the number of milliseconds the point value must have been beyond or equal to the limit value to enter the corresponding new state. The enter persistence is only used when a new limit state is entered with the point value moving away from the normal state.

2.4.5.7 Leave persistence

The leave persistence defines the number of milliseconds the point value must have been beyond the limit value with hysteresis to leave the current state and enter another state. The leave persistence is only used when a new limit state is entered with the point value moving towards the normal state.

2.4.5.8 Limit profile number

The Limit Profile numbers define the profiles to be used for the limit. This allows for different limits to be applied at different times of the day and on different days of the week. The profiles must have floating point values, and the FD rejects any configuration with a different type of value. For a fixed limit, no profile, use profile number zero.

2.4.6 Counter limits

Counters may have limits. When the counter exceeds those limits, events can be raised. It is possible to define multiple limits on counters. To configure multiple limits on the same point, the configuration must contain multiple limit configuration elements. When an SA is configuring a sub-set of the limits on a point, it must ensure they do not invalidate existing limits. An SA may decide to download all of the limits for a point even if some of the limits remain unchanged.

The following rule applies to all enabled limits for an individual counter point. Disabled limits are ignored and need not be present on upload.

NOTE:	**The limits must be set in ascending order of limit value with PLI otherwise the FD rejects the configuration. When a profile is being used every day of the week there is no static limit value being used. In this case the unused limit value can contain any arbitrary value and must not be included in any validation checks done in the FD.**

The state of a counter point corresponds to the region between two adjacent limits, or the region above the highest limit. The region between 0 and the lowest limit is referred to as the normal state. The range of values represented by a particular state includes the value of the lower bound, but not the value of the upper bound. A point without any limits is always in the normal state, see Figure 10.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

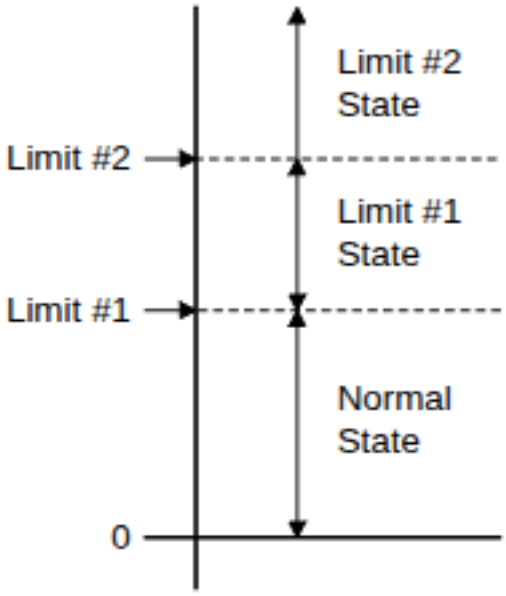


Figure 10: Counter normal state

For example, given a point with two limits at 20.0, high, and 40.0, high-high, the normal state represents values less than 20.0, the high state represents values greater than or equal to 20.0 and less than 40.0, and the high-high state represents values greater than or equal to 40.0.

The information in Table 15 is used to define limits for counter points within the object: **counterLimit**.

Table 15: Information used in defining a counter limit

Information	**Type**	**JSON Name**	**Mandatory**	**Description**
Parent Point	Integer	parent	Yes	Counter point number.
Point Limit Index	Integer	num	Yes	Point Limit Index (PLI) for limit.
Limit Name	String	name	No	A human readable name for the limit.
Action	Integer	action	Yes	Action to be taken on entering this state.
Limit Value	Number	value	Yes	The value of this limit.
Monday Profile	Integer	mon	No	Profile number for Monday.
Tuesday Profile	Integer	tue	No	Profile number for Tuesday.
Wednesday Profile	Integer	wed	No	Profile number for Wednesday.
Thursday Profile	Integer	thu	No	Profile number for Thursday.
Friday Profile	Integer	fri	No	Profile number for Friday.
Saturday Profile	Integer	sat	No	Profile number for Saturday.
Sunday Profile	Integer	sun	No	Profile number for Sunday.

If a counter point is cleared to 0 then the point enters the normal state.

When a counter point changes state, only the new state of the point is reported. Any intermediate states that may have been entered in the change from the old state to the new state are not reported. This ensures that only a single event is generated, or event logged for each state change. For example, if a point has two high limits and changes from the high-high state to the normal state, after being cleared to zero, then one data set must be generated, or event logged for entering the normal state.

When multiple changes occur in one scan of I/O for a point, state, point quality flag, or other changes, only the highest action of all the changes that occurred is performed by the FD subject to the highest permitted action. This is to ensure only one action occurs for a single I/O scan.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

If counter rollover occurs, because of a point incrementing beyond the maximum possible counter value, then the FD must not generate an event, or log an event as a consequence of the rollover. Also, following a rollover the FD must not generate any further events or log any further events until the counter is cleared to 0.

2.4.6.1 Point limit index

The PLI defines which of the point's limits the record relates to and must be a valid limit for the FD.

The PLIs used by the FD must start from one and be consecutive. Also, PLI one must correspond to the lowest limit, and the maximum PLI must correspond to the highest limit. PLI zero is reserved to represent the normal state of a point. A maximum number of limits for each counter point is declared in the SA Profile.

For example, an FD with two limits may use the following PLIs:

- 1 – High
- 2 – High-High

2.4.6.2 Limit value

The limit value defines the threshold for the limit.

When a profile is specified for a day of the week, it overrides the limit value for that day. When a profile is being used every day of the week then the limit value is not used and can contain any arbitrary value. In this case the FD must not perform any validation using the limit value.

2.4.6.3 Action

The Action defines the action to be taken on exceeding the limit in either direction. The action is as defined in Section 2.6, Limit and control profiles.

When a point changes state the action that is performed by the FD is determined by the highest configured action of the enabled limits that were exceeded as the result of the change:

- If any of these actions are configured as value 2, event and request connection, then the FD must create an event for the new state and assert **request to connect**, so the SA may raise or clear the required alarm(s).
- If none of these actions are configured as value 2, but any of them are configured as value 1, event, then the FD must store an event for the new state.
- Only if all these actions are configured as value 0, do nothing, will the FD perform no action.

When a point is on scan and any of the point quality flags change, the action for the flag must be performed subject to the highest permitted action, to report the new object flags. However, this does not apply until the Restart Point quality flag is cleared, and the state of the point determined.

In the following situations the highest action of all of the enabled limits and all of the point quality flags must be performed regardless of the highest permitted action. This highest action must also include any action configured for significant change of value reporting. This is required to report the current state of the point.

- At device start up and following a restart.
- When the point is put on scan.
- Following a configuration change to the action for any limit or any of the point quality flags.
- When a change in the configuration of the point's limits causes the point to change state.

The action is performed once the point has completed the next or initial scan of its I/O, that is, when the Restart Point quality flag is cleared, without applying hysteresis or persistence.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

2.4.6.4 Profile number

The profile numbers define the profiles to be used for the limit, to allow different limits to be applied at different times of the day and on different days of the week. The profiles must have floating point values, and the FD rejects any configuration with a different type of value. For a fixed limit, no profile, use profile number zero.

2.4.7 No change

Analogue and counter points support no change detection. The change in value over a period of time is calculated and then compared to a deadband.

No change detection is performed on the scaled value of the point.

The information in Table 16 is used to define no change for analogue and counter points within the object: **noChange**.

Table 16: Information used in defining no change for analogue and counter points

Information	Type	JSON Name	Mandatory	Description
Parent Point	Integer	parent	Yes	Analogue point number.
Parent Point Type	String	parentType	Yes	Type of point the limit refers to, “ai”, “ao”, “x”, “c”.
Point Limit Index	Integer	num	Yes	Point Limit Index (PLI) for limit.
Name	String	name	No	A human readable name for the limit.
No Change Deadband	Number	deadband	No	The size of the deadband within which the point value must remain to cause the no change event. Default 0.
No Change Period	Integer	period	Yes	The time in milliseconds that must elapse with the value remaining within the deadband to cause the no change event (0=disable no change detection).
No Change Action	Integer	action	Yes	Action to be taken on no change (0=do nothing, 1=log event with reason, 2=log event and trigger a connection to the SA).

The no change deadband is the non-negative value that defines the maximum amount by which the point’s value can change within the change period and still be deemed to have not changed.

The no change deadband is specified in engineering units, and the FD must store it in engineer units, not just the equivalent raw value, so that it is not affected by any subsequent change to the scaling of the point. A no change deadband of 0 is valid and means that any non-zero change indicates the point’s value has changed.

The no change period is the time in milliseconds that the point must remain within the deadband to cause the event to occur. A no change period of 0 turns off no change detection.

The period is started from the time the point value is first read and is reset when a point exceeds the no change threshold. At this time the value which is used to compare current values is reset to the changed value.

If a counter point rolls over during a no change period, then the FD must compensate when calculating the no change indication. The following algorithm is used to calculate if the value has not changed:

abs (new value - old value) ≤ no change deadband

Only valid values are included in the derivation. The source point is deemed valid if the Offline Point quality flag is clear, and the Restart and Comms lost flags are both clear. However, if the Local Forced flag is set, the point should convey valid data and may be considered in the calculation regardless of

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

the other flags. The Remote Forced flag may also indicate valid data provided that Offline is clear and Comms Lost is clear. Other combinations of the point quality flags indicate that the data is invalid.

The action to be taken is as described in Section 2.6, Limit and control profiles.

2.5 Extended points

The protocol supports extended point types. An extended point is calculated from information taken from a source point.

Source points can be any binary, analogue or counter input, or binary or analogue output.

An extended point may present statistical, summary or event-based information for the source point. Apart from their method of calculation and how their point quality flags are set, extended points behave as for other points. They may be logged or have alarm limits associated with them. Rules applicable to extended point types are given below:

- For minimum, maximum, and mean, the stored value is the calculated value for the most recently completed calculation period. Therefore, the value in the extended point is only updated at the end of each period. This value then remains in the extended point until the end of the next period.
- For integral, state counter, and state runtime, the stored value is updated during the calculation period with the value being reset to 0 at the start of the next calculation period.
- Calculation is performed on source point values during the calculation period. These values are taken at the underlying default update rate for the source point or whenever logged values are requested for the source point.
- To remove an extended point, or stop calculation of the point, a full configuration which does not include the point is required.
- Only valid values are included in the calculation. The source point is deemed valid if the Offline Point quality flag is clear, and the Restart and Comms Lost flags are both clear. However, if the Local Forced flag is set, the point should convey valid data and may be considered in the calculation regardless of the other flags. The Remote Forced flag also indicates valid data provided that Offline is clear and Comms Lost is clear. Other combinations of these point quality flags indicate that the data is invalid.
- For minimum, maximum, and mean the point quality flags Range Error and Reference Error are a snapshot of the corresponding flags of the source point at the end of the time period, or at the time the event defining the extended point Max or Min occurred.
- For integral, state counter, and state runtime the point quality flags Range Error and Reference Error are the logical OR of the corresponding flags of the source point for all individual samples within the period.
- The extended point always has the Offline flag clear unless the source point was invalid for the entire sampling period.
- The Local Forced flag on the extended point indicates that the extended point itself has been over-ridden. The Remote Forced flag on the extended point is never set.
- When an extended point configuration is processed by the FD it must determine if there is a change being made to the configuration of the extended point's index. For any change of index, it should set the value of the extended point to the current value of the source point and set the extended point's Restart quality flag. This ensures that the value and flags set from any previous use of the point are not interpreted as pertaining to the newly calculated value.
- If the extended point is configured for periodic logging and the log period coincides with the end of a calculation period, then the FD should calculate the new value before the logging takes place. This is to ensure that the logged value is pertinent to the log period.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

- To calculate an extended value of a point over an additional time period, the extended point used to store the initial extended value may itself have a further calculation provided the second calculation is synchronised with the first.

The information in Table 17 is used to define extended points within the object: **extPoint**.

Table 17: Information used in defining extended points

Information	Type	JSON Name	Mandatory	Description
Parent Point	Integer	parent	Yes	Parent point number.
Parent Point Type	String	parentType	Yes	Type of parent point the extension refers to, “ai”, “bi”, ”ao”, “bo”, ”x”, ”c”.
Extension Point Index	Integer	num	Yes	Extension point number (allowable numbers defined in the min/max/totalObjects).
Name	String	name	No	A human readable name for the point.
Class Name	String	className	No	The class is optional and is a string which can be used by the SA to denote equipment or asset ontology.
Extension Point Type	String	type	Yes	One of: “RoC” (Rate of Change), “Min”, “Max”, “Mean”, “StdDev”, “Integral”, “Run Time”, “Count” Run times count in 1 second units, scaled by the scaleFactorType of the extension. Counters count 1 per value change to ‘state’ value, also scaled.
Reset Period	integer	resetPeriod	No	Interval between resets of this extension’s value calculation, in milliseconds). If zero or absent, then is not reset.
Reset Offset	integer	resetOffset	No	Offset from start of week (Monday) in milliseconds.
Report Period	integer	reportPeriod	No	Interval between reports of this extension’s value. In milliseconds, starting at reset with last accumulated value.
Scale Value	number	scale	No	Multiplies raw value to get value of extension.
State Mask	integer	stateMask	No	Increment the Run Time or Count when the source point state matches a state set in this bit mask. Only applies to Run Time or Count. Bit 0 is state 0, bit 1 state 1 and so on. A state mask of 0 will result in no Run Time or Count updates.
Units	string	units	No	The units are a free format ASCII string that can be used when displaying the point value through local access displays etc. The FD should not interpret any meaning from this string and may choose to ignore it completely.

The Report and Reset values are indicated by the relevant Reason value in the reported data.

2.5.1 Minimum

The minimum extended point is an analogue input which captures the minimum value of a source analogue point over a given time period.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

The reset period is the time in milliseconds over which the minimum is calculated.

The offset is the number of milliseconds from the start of the week to which the reset period must be synchronised.

The report period is the time in milliseconds between logging values. The minimum value may therefore go lower in successive reports until reset.

2.5.2 Maximum

The maximum extended point is an analogue input which captures the maximum value of a source analogue point over a given time period.

The reset period is the time in milliseconds over which the maximum is calculated.

The offset is the number of milliseconds from the start of the week to which the reset period must be synchronised.

The report period is the time in milliseconds between logging values. The maximum value may therefore go higher in successive reports until reset.

2.5.3 Mean

The mean extended point is an analogue input which captures the mean value of a source analogue point over a given time period.

The reset period is the time in milliseconds over which the mean is calculated.

The offset is the number of milliseconds from the start of the week to which the reset period must be synchronised.

The report period is the time in milliseconds between logging values. The mean value is therefore calculated from the reset time to the report time.

2.5.4 Standard deviation

The standard deviation extended point is an analogue input which calculates the Population standard deviation of the input.

The reset period is the time in milliseconds over which the deviation is calculated.

The offset is the number of milliseconds from the start of the week to which the standard deviation reset period must be synchronised.

The report period is not used.

2.5.5 Integral

The integral extended point is an analogue input which captures the integrated value of a source analogue point over a given time period.

The reset period is the time in milliseconds over which the integral is calculated.

The offset is the number of milliseconds from the start of the week to which the integral reset period must be synchronised.

The integral is the sum of a series of analogue values, each multiplied by the elapsed time (in seconds) since the preceding value.

The report period is the time in milliseconds between logging values. The integral value is therefore calculated from the reset time to the report time.

The analogue values used when calculating the integral should be chosen in such a way that they give the most accurate result for the particular FD and I/O.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

For example, if an FD periodically samples the current value from its I/O then it can use the mid-value (average) between consecutive samples.

In Figure 11, the integral is calculated over a period of 5 minutes.

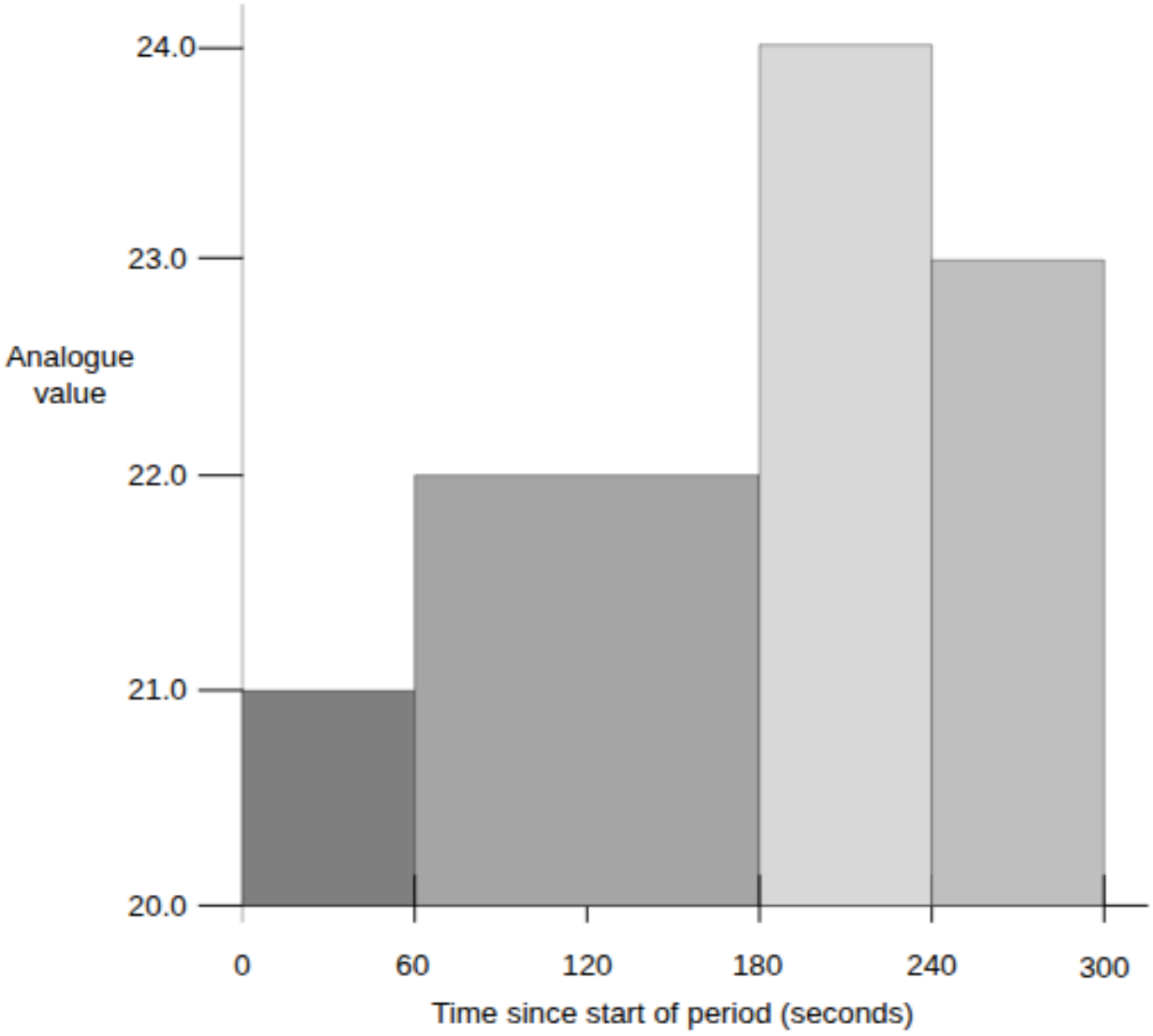


Figure 11: Example of calculation of integral extended point value

Using the analogue values shown in the example the integral would be as follows:

(21.0 × 60) + (22.0 × 120) + (24.0 × 60) + (23.0 × 60) = 6720.0

The value stored in the virtual point is the integral since the start of the current calculation period. At the start of the next reset period, this value is reset to zero.

2.5.6 Rate of change

Analogue and counter points may support Rate of Change (ROC) detection. The ROC is calculated over a period of time and then scaled to a specified time base with the results stored in an extended analogue input point, which can then have events configured.

ROC detection is performed on the scaled value of the source point.

The ROC reset period is the time in milliseconds that the point must remain within the deadband to cause the event to occur. An ROC period of 0 turns off ROC detection.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

The ROC timebase is the number of milliseconds over which the ROC is scaled in the calculation.

The ROC point number is the index of an extended analogue input that is used only to store the calculated ROC of the analogue or counter point. If a counter point rolls over during a ROC period, then the FD must compensate when calculating the ROC. The following algorithm is used to calculate the ROC:

$$\frac{new\,value\ -\ old\,value}{ROC\,period}\cdot\ ROC\,timebase$$

Where *new value* is the current value of the point and *old value* was the value of the Point *ROC period* milliseconds earlier. For example:

$$\frac{12.0\,mm\ -\ 9.0\,mm}{30,000\,ms}\cdot 60,000\,ms =\ Rising\,by\,6.0\,mm\,per\,minute$$

The ROC is negative if the value is falling, positive if the value is rising and zero if unchanged over the defined period.

Only valid values are included in the derivation. The source point is deemed valid if the Offline Point quality flag is clear, and the Restart and Comms lost flags are both clear. However, if the Local Forced flag is set, the point should convey valid data and may be considered in the calculation regardless of the other flags. The Remote Forced flag may also indicate valid data provided that Offline is clear and Comms Lost is clear. Other combinations of the point quality flags indicate that the data is invalid.

No attempt is made to capture any of the source point quality flags into the extended point quality flags.

The ROC extended point always has the Offline Point Quality flag clear if the source point was valid at the start and end of the sampling period.

The Local Forced flag on the ROC derived point indicates that the extended point itself has been overridden. The Remote Forced flag on the extended point is never set.

When ROC configuration is processed by the FD it must determine if there is a change being made to the configuration of the extended point. For any change of the ROC extended point number it should set the value of the extended point to zero and assert the extended point's Restart quality flag. This ensures that the value/state and flags set from any previous use of the points are not interpreted as pertaining to the ROC value indication.

If the ROC extended point is configured for periodic logging and the log period coincides with the end of a calculation period, then the FD should calculate the new value before the logging takes place. This is to ensure that the logged value is pertinent to the log period.

The ROC periods should be synchronised to the reset offset.

The action to be taken is as described in Section 2.6, Limit and control profiles.

The report period is not used.

2.5.7 State counter

Binary, analogue, and counter points all support state change counting. The state counter extended point is a count of the number of times the point's state has been changed to one of a selected set of states within a given time period. The count is stored in the state counter extended analogue point.

The Type field is **Count**.

The reset period is the time in milliseconds over which the count is calculated.

The reset offset is the number of milliseconds from the start of the week to which the count period must be synchronised.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

The state mask defines which states of the point are to be counted. The mask has 1-bit per state, with bit 0 corresponding to state 0, bit 1 to state 1 and so on. For example, a state mask of 0x0005 counts the number of times a double-bit binary input point changes to state 0 or state 2 within the defined period.

To disable the state counter on the point, specify a state mask of zero.

The extended point value is the number of times the point enters one of the selected states within the given time period.

The report period is the time in milliseconds between logging values. The counter value is therefore calculated from the reset time to the report time.

2.5.8 State runtime

Binary, analogue, and counter points all support state runtime calculation. The state runtime extended point is time within a given time period that a point was in one of a selected set of states. The time is stored in the state runtime extended analogue point.

The Type field is **Run Time**.

The reset period is the time in milliseconds over which the time is calculated.

The reset offset is the number of milliseconds from the start of the week to which the runtime period must be synchronised.

The state mask defines which states of the point are to be included in the count of time. The mask has 1-bit per state, with bit 0 corresponding to state 0, bit 1 to state 1 and so on. For example, a state mask of 0x0005 calculates the time that a double-bit binary input point has been in state 0 or state 2 within the defined period.

To disable the state runtime on the point, specify a state mask of zero.

The extended point value is the time in milliseconds that the point has been in one of the selected states within the given time period.

The report period is the time in milliseconds between logging values. The runtime value is therefore calculated from the reset time to the report time.

2.6 Limit and control profiles

Profiles are time-varying values. A profile can be attached to an analogue or counter limit to cause that limit to vary across a week. They can also be attached to a control profile object.

2.6.1 Limit profiles

A Limit Profile can be attached to an analogue or counter alarm limit or be attached and used by a control profile.

The information in Table 18 is used to define limit profiles within the object: **limitProfile**.

Table 18: Information used in defining limit profiles for analogue and counter points

Information	Type	JSON Name	Mandatory	Description
Limit Profile Index	Integer	num	Yes	An index for this limit.
Values	Array of Numbers	values	Yes	Values array - first value applies at first time.
Times	Array of Integer	times	Yes	Times array - in milliseconds from start of day.
Type	Integer	type	No	Whether the profile is stepped (0) or linear (1). Stepped is 'step first'. Stepped is the default.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

Information	Type	JSON Name	Mandatory	Description
Interpolate	Integer	inter	No	The interpolation time (in milliseconds) is the time period used to compute a new value for the profile, interpolating between two profile points. If set to 0 then no interpolation is performed. Applies to control values.

2.6.2 Control profiles

The information in Table 19 is used to define limit profiles within the object: **controlProfile**.

Table 19: Information used in defining a control profile

Information	Type	JSON Name	Mandatory	Description
Parent Point	Integer	parent	Yes	Analogue or binary output point number.
Parent Type	String	parentType	Yes	“ao” or “bo”
Control Profile Index	Integer	num	Yes	Control Profile Index for this profile.
Limit Name	String	name	No	A human readable name for the limit.
Monday Profile	Integer	mon	No	Profile number for Monday.
Tuesday Profile	Integer	tue	No	Profile number for Tuesday.
Wednesday Profile	Integer	wed	No	Profile number for Wednesday.
Thursday Profile	Integer	thu	No	Profile number for Thursday.
Friday Profile	Integer	fri	No	Profile number for Friday.
Saturday Profile	Integer	sat	No	Profile number for Saturday.
Sunday Profile	Integer	sun	No	Profile number for Sunday.

The Interpolate field in the Limit Profile determines how often a value is set on the control output.

2.7 Calculated points

Calculated points are linked to a Device Link object with a specific name Calc. They are calculated by the FD using a numeric set of codes to determine the formula. The time and offset of calculation are determined by the scanRate and scanOffset. The codes and other fields are specified in Table 20 as additional items added to the object type: **point**.

Table 20: Information used in defining a point calculation

Information	Type	JSON Name	Mandatory	Description
Calc Code Data	Array of Numbers	calcCode	No	Specify this only for a calculated point (the deviceLink is ”Calc”). The code is compiled from an expression by the configuring application.
Calculate On Change	Boolean	calcOnChange	No	Causes calculation when any input values or their qualities change.

For details on the calculation codes and its use, please refer to Appendix C. Calculated points are based on the core point properties and therefore support limits, extended points, and incident logging. The quality flags also apply with some additional notes in the appendix concerning expression or calculation errors. Calculations refer to other points by an enumeration of the point type and their point number.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

2.8 Incident logging

In most applications the logging behaviour of points and extension points, as determined by their configuration, is sufficient. There are some added logging facilities provided by the incident logging object.

The incident logging object defines additional history data logging using an object which is linked to the point to be logged. Generally, the logging of the additional history is continuous but only selectively sent to the SA.

There are two circumstances where the logging is sent:

- On request by a data retrieval message from the SA
- When the value of a second “trigger” point meets a given condition

The information in Table 21 is used to define an incident log within the object: **incidentLog:**

Table 21: Information used in defining an incident log

Information	Type	JSON Name	Mandatory	Description
Parent Point	Integer	parent	Yes	Parent point number.
Parent Type	String	parentType	Yes	Type of the source point “ai”, “bi”, “ao”, “bo”, “c” or “x”.
Store Number	Integer	num	Yes	Index number for this profile.
Trigger Point	String	triggerPoint	No	Number of the point causing this log to be delivered.
TriggerType	String	triggerType	No	Type of the trigger point “ai”, “bi”, “ao”, “bo”, “c”.
Trigger Mask	Integer	triggerMask	No	Trigger occurs when point is in a state matching this bit mask. Binary inputs match by value. Analogues and counters match by alarm limit number, where the limit numbers correspond to bit numbers.
Trigger Start	Integer	triggerStart	No	The time (relative to the start of the incident) when logging is to begin (in milliseconds). Negative values mean the log starts before the trigger time.
Trigger Length	Integer	triggerLength	No	The length of time (in milliseconds) to continue logging the point after the incident.
Action	Integer	action	No	Action to be taken on entering this state (0=do nothing [not relevant in this object type], 1=log event with reason, 2=log event and trigger a connection to the SA).
Log Rate	Integer	logRate	Yes	The rate to be used (in milliseconds) to log the point.
Log Offset	Integer	logOffset	No	Offset time from the start of the week that the log is synchronised from (Monday, 00:00:00). Measured in seconds. If 0, then log is offset from the last midnight. (Default 0).

Where no Trigger fields are present, the FD logs at the rate specified, and offset if specified, and the retrieval of the log is caused by a data message request from the SA.

Where Trigger fields are used, they cause the selected data to be sent unsolicited from the FD to SA. The action field controls whether the FD immediately contacts the SA or waits for the next connection.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

3 Controls and overrides

This section provides information on the controls and overrides used in the protocol.

3.1 Controls

NOTE:	Applies to analogue output, binary output, and string points only.

If supported as listed in the features section of the profile, the SA may send a message to the FD to control a point output. The evidence of an output can be received through the value received in the data message, as configured on the output point.

[UUID]/DOWN/control

The QoS is 2, and the message is:

- Control by point identifications:

```
{
    "c":[{"ao":1,"v":1}]
}
```

- Or by name:

```
{
    "c":[{"name":"Pump 1.Command","v":1}]
}
```

- If the value is a string:

```
{
    "c":[{"name":"Pump 1.State","vs":"Fail"}]
}
```

Use the value property “v” only for numeric point types (Analog and Binary) and “vs” only for the string point type.

If this is a pulsed control, that is, the 1 bit binary output has a non-zero pulse duration, the value is the time in milliseconds which overrides the value specified in its configuration.

If a point attached to the “Internal” Device Link is to be modified by controls, then it must be typed as an analogue output or binary output.

3.2 Overriding a point

Subject to the **canOverride** property of points.

Extensions cannot be overridden.

When overridden the point quality is set accordingly.

If supported as listed in the features section of the profile, the SA may send a message to the FD to override the value of an input point.

[UUID]/DOWN/override

The QoS is 2, and the messages are:

- Override by point identifications:

```
{
    "o":[{"ai":1,"v":1}]
}
```

- Or by name:

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

```
{
    "o":[{"name":"Pump 1.Rate","v":1}]
}
```

- If the value is a string:

```
{
    "o":[{"ai":1,"vs":"test value"}]
}
```

- Release by point identifications:

```
{
    "r":[{"ai":1}]
}
```

- Or by name:

```
{
    "r":[{"name":"Pump 1.Rate"}]
}
```

Use the value property ”v” only for numeric point types (Analog, Counter and Binary) and ”vs” only for the string point type.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

4 Data reporting

The FD remembers what data has not previously been sent. That data is then fully transmitted on a connection, for example:

- On scheduled connection
- On an action which causes a connection
- On receipt of a request (as in following section)

Unsent data includes time-logged or event data.

4.1 Data formats

The FD sends data values to the SA in a flexible data format. This supports simple data as well as full detail. While in JSON format, variable names and data layout are optimised to reduce message size.

Data is sent in the order that it is gathered. FD Configuration regulates when and how data is sent, such as on a priority basis. However, earlier data is always sent before later data unless the FD is responding to a request for data gathered in a specified period.

Reporting of data is always in ascending time order unless either of these two boolean properties are true in the FD Profile.

- If reportEventsDirectly is specified with true, then the FD may send data values where the Action type is 3, and the data values associated with that event can be sent before other data values.
- If respondCurrentDataDirectly is specified with true, then the FD will respond to SA current data request messages with the latest current data before sending older data.

[UUID]/UP/data

Points can be referred to by their name or by a specific point type and numeric index ":".

Valid point types are defined in table 9.

The numeric format will reduce message size and may be preferred by field devices.

Note that it is valid to include both name and type / index, but if are included then only the type / index element is used, the 'name' is ignored.

The QoS is 1. The JSON message is:

```
 // All data in the block:
{
 // version of the configuration settings data
"cVer":123,
 // All data for this point
"p":[
  {
 // Identification of this point, either
 // by point type and point number:
   "ai":1,
 // or by name:
   "name":"Pump 1.Rate",
 // Value as real or integer (true or false are not
 // used) for analogue, counter or binary points:
   "v":[12.3, ...],
 // Or the value may be a string value for text or
 // BASE64 MIME data in string points only.
   "vs":["string content", ...],
 // Optional quality flags as bitmask, date/time of this
 // value. This can include the override flag. Default
```

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

```
 // quality is 0.
    "q":[1, ...],
 // Time of start and interval in milliseconds for
 // subsequent values,
    "t":{"s": 1700497254000,"i":900000},
 // Or individual time values as an array.
    "ts":[ 1700497254000, ...],

 // Optional if the reason is 4 (Timed Report)
 // reasons for logging the value:
 // 0="No Change",
 // 1="Current",
 // 2="Value Change",
 // 3="State Change (i.e., Alarm)",
 // 4="Timed Report (log)",
 // 5="End of Period",
 // 6="End of Period Reset"
 // Default if not specified is 4.
    "r":[4, ...],
 // Optional alarm state (default is 0),
 // -1 is a general alarm indication, 1..2..3 etc are
 // alarm limit numbers.
 // 100 and 101 are under-range and over-range
 // respectively.
    "a":[1, ...]
    }
  ]
}
```

Only Alarm, Quality and Reason can be optional, and these have default values.

All array sizes must match with the following exception. Any of the Alarm, Quality or Reason items can be an array containing a single value which then applies to all values in the block for that point.

All data blocks are processed from start to end. Data for a single point must be sent in chronological order, although data for multiple points can be interleaved or in point order.

Data messages carry the FD's configuration version.

Note: If the version within the data message differs from the configuration currently active within the SA, then the data must be discarded (or otherwise excluded from processing).

Implementation Note / Future Consideration for SA's

The default requirement is to for the SA to discard inconsistent messages. However, an SA may elect to provide more flexible SA specific optional behaviour and buffer these inconsistent data messages until the new configuration has been received, validated, and activated by the SA. Once alignment is confirmed, the buffered data could then be processed using the updated configuration mappings, preventing data loss during configuration transitions.

4.2 Requesting data

If supported as listed in the features section of the profile, the SA may send a message to the FD to request data values. This is optional and a device may be designed only to report data.

Typically, a device stores data until its scheduled time to connect and send data, or until a high priority event, sometimes termed alarm, causes connection and send.

[UUID]/DOWN/getdata

The QoS is 0, and the message is:

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

- **Or** to retrieve a list of current point values, one for each configured point. To avoid issues with data ordering the device sends stored/unsent data before sending the current point data for all points, including timestamps. The Current Data Reason flag is used for these values.

```
{"data":"current"}
```

- **Or** to retrieve a list of current (at this time) point data, by point type / index. Again, to avoid issues with data ordering, the device sends all unsent data before then sending current point data.

```
{
    "data":"current",
    "p":[{"ai":1},{"ai":2}]
}
```

- **Or**:

```
{
    "data":"current",
    "name":["Pump 1.Rate"]
}
```

With an optional time start to (re-)retrieve historic data from this time to the current time. Historic values include data stored because of time Incident Logging:

```
{
    "data":"history",
    "p":[{"a":1}]
}
```

- **Or**:

```
{
    "data":"history",
    "name":["Pump 1.Rate"],
    "start":1694695035000,
    "end":1694695235000
}
```

The end time is optional.

The expected response is a standard data message.

Table 22: Elements of the data request message

Information	Type	JSON Name	Mandatory	Description
Type of Data Requested	String	data	Yes	“Current” or “History”
Point ID List	String Array	n	No	Array of point identifiers
Point Name List	String Array	name	No	Array of point names
Start Time	Integer	start	No	Start time (inclusive)
End Time	Integer	end	No	End time (inclusive)

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

5 Field device core blocks

These are one or more data blocks or files which can optionally be uploaded and downloaded to set manufacturer-specific configurations and behaviours. See **canCoreUp** and **canCoreDown** in the FD Profile.

These are typically downloaded to the device either because the SA is instructed by users to do so, or the Required flag is set in **core** in the status message.

When successfully downloaded to the FD, the status message is updated with the new version. This indicates that the download is successful.

[UUID]/DOWN/core/block/<name>/<version-text>

The <name> is the text name given in the core structure. It could be Firmware. After a Core block is transmitted in the message content, the SA can, at that time or later, request that the block is made active. This typically applies to a logic sequence. The command is:

[UUID]/DOWN/core/request

With content:

```
{ "activate" : ["<name1>", "<name2>"] }
```

And to deactivate, use the same topic with content (if applicable, e.g. to Logic programs):

```
{ "deactivate" : ["<name1>", "<name2>"] }
```

If the SA requires that a core block is uploaded (if supported by the device), the request is:

[UUID]/DOWN/core/request

With content:

```
{ "upload" : ["name1", "name2"] }
```

And the device responds with the content topic:

[UUID]/UP/core/block/<name>/<version-text>

And the data in the message content.

The core block data format is.

```
{ "blockPart" : 1, "totalParts" : 1, "data" : "TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24
```

The data is BASE64 encoded. This allows the block to be composed of multiple messages, by dividing the block into multiple parts. A block is only complete with all parts downloaded/uploaded. The JSON message may be compressed, see Section 6.3.1, Payload compression.

NOTE:	This may be necessary for transports which support limited message sizes.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

6 Use of MQTT

MQTT is a standards-based messaging protocol, or set of rules, used for machine-to-machine communication. Internet of Things (IoT) devices typically have to transmit and receive data over a resource-constrained network with limited bandwidth. These IoT devices use MQTT for data transmission and can communicate IoT data efficiently. MQTT supports messaging between devices to the cloud and the cloud to the device. This section provides information on the use of MQTT within the Lucid protocol.

Lucid supports MQTT versions 3.1.1 and 5.0.

6.1 Retained messages and connection state

The protocol takes advantage of Retained messages for Configuration and Profile.

A vendor may choose to use the Clean Session or Clean Start flags for the SA or FD. The responsibility for ensuring that subscriptions are in place lies with the SA or FD individually, and the FD may need to avoid using these flags if connected periodically.

A vendor may choose whether to configure the FD so that the SA is aware, or unaware, of its connection state.

Where the FD requires the SA to be unaware of the connection state, the LWT message is optional. If the LWT is used, the SA can raise alarms or events when an unsolicited disconnection occurs. The FD sends an MQTT Disconnect message to disconnect.

Where the FD requires the SA to be aware of connection state, the LWT message is mandatory. The FD does not send an MQTT Disconnect message.

6.2 Publish/subscribe transports

The MQTT transport requires a message Broker between the SA and the FD. The SA and the FD subscribe to Topics so that the Broker can send messages to the correct destinations.

The MQTT Broker maintains persistent stores for every FD that publishes information set as **Retain**. This includes the FD Profile and its Configuration. It is assumed that Retain messages are kept indefinitely by the MQTT Broker until deleted or overwritten by the FD.

6.3 Payload

Settings and data are encoded and transferred within JSON messages between the SA and FD. The JSON is in UTF-8 with no Byte Order Mark (BOM).

Packet sizes are restricted by the MQTT protocol. The SA is expected to handle all packets sent by the FD, and the FD is expected to handle any packet size it requires for its own functions.

NOTE:	**This document gives JSON code, and for explanatory purposes includes comments within the code after // characters. These comments MUST NOT be included within any messages. Note also that JSON messages SHOULD not contain extraneous spaces or line breaks, but devices and SA's MUST tolerate them.**

6.3.1 Payload compression

Compression of all data in a message can be used by the FD and may be supported by the SA if indicated in the SA Profile. This is optional. The names of compression algorithms are specified case insensitive in the Profiles.

Compressed payload messages are indicated with a prefix of two bytes, '$' (hex 0x24, decimal 36) followed by a compression-specific byte ranging from 1 upwards.

Algorithms supported are:

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

- GZip compression, “gzip”. Second byte is ‘1’ (hex 0x31, decimal 49).
- Brotli compression, “br”. Second byte is ‘2’ (hex 0x32, decimal 50).
- Deflate compression, “deflate”. Second byte is ‘3’ (hex 0x33, decimal 51).

These bytes are followed by the compressed payload.

6.4 Other topics

Messages from the SA to the FD are stored in Topics that start with [UUID]/DOWN/ where [UUID] is the UUID of the field device that the message is destined for. Messages from the FD to the SA are published to topics starting with [UUID]/UP/, where [UUID] is again the UUID of the FD.

NOTE:	**The FD does not need to know the identity of the SA (or Stations) that receive messages from the FD.**

6.4.1 Getdeviceinfo

If supported as listed in the features section of the FD Profile, the SA may send a message to the FD to request information. This is optional.

An SA may request different elements. The responses are as defined for each of the elements above. Responses are sent on their respective Topics. There are no time constraints on the response, and other messages could be sent first.

[UUID]/DOWN/getdeviceinfo

The QoS is 0, and the message is:

- Cause status message in the FD to be sent:

```
{"config":"status"}
```

- Cause FD Profile data in the FD to be sent (if capable):

```
{"config":"profile"}
```

- Cause configuration data in the FD to be sent.

```
{"config":"configuration"}
```

6.4.2 Maintaining time

Where there is no better alternative to setting the time on an FD, this optional part of the protocol provides a simple but low accuracy method.

If supported by both SA and FD (as specified in the FD Profile and SA Profile), the SA may send a message to the FD to set its clock. Alternatively, the device may request the SA time. Support is to be indicated in the FD Profile data of SA and FD.

This method uses an exchange of messages to gather a time difference between FD and SA and allow the FD to set the time if appropriate. If the FD requires its clock to be set, it can indicate this in the status message. The SA is responsible for determining when the clock is to be set, whether this is to a timed schedule or not.

There are two ways to set time described in these two following sections:

6.4.2.1 From the status message

When the SA receives a status message which contains the time and the timeReq value is true, the SA responds by sending a setclock message.

The FD retains the last sent time in the status message. The FD can verify its own sent time was received and then set its own clock, if satisfied that the get/set interval indicates the clock can be set reliably.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

[UUID]/DOWN/setclock

```
{"sa":1694695039000,
"fd":1694695035000}
```

6.4.2.2 From an SA request

If the SA is to set the clock independently of receiving a status message, it sends:

[UUID]/DOWN/setclock

The QoS is 0, and the message is the SA's current time:

```
{"sa":1694695032000}
```

The FD may receive this message at some time later, and so it would not use this time. It responds with its own clock request as below.

[UUID]/UP/getclock

With the content:

```
{"sa":1694695032000,
"fd":1694695035000}
```

The FD uses the received SA's time within this message. If the SA receives this message and identifies the same SA time within the message, it repeats its setclock message with a new time and the last time sent by the FD. The FD can then verify its own sent time was received and then set its own clock, if satisfied that the get/set interval indicates the clock can be set reliably.

[UUID]/DOWN/setclock

```
{"sa":1694695039000,
"fd":1694695035000}
```

In each of the above two cases the SA and FD each use their own time tolerance figures to determine whether the times are:

a. Different enough to require a change.

b. Received in a short time which indicates both parties and communicating in near real-time. What constitutes a sufficiently short time is implementation-specific.

6.5 Communications, QoS, retention

Refer to Appendix B for a summary by topic of the names, quality of service and retention attributes to be used.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

7 Field device lifecycle

Figure 12 shows the connection state model.

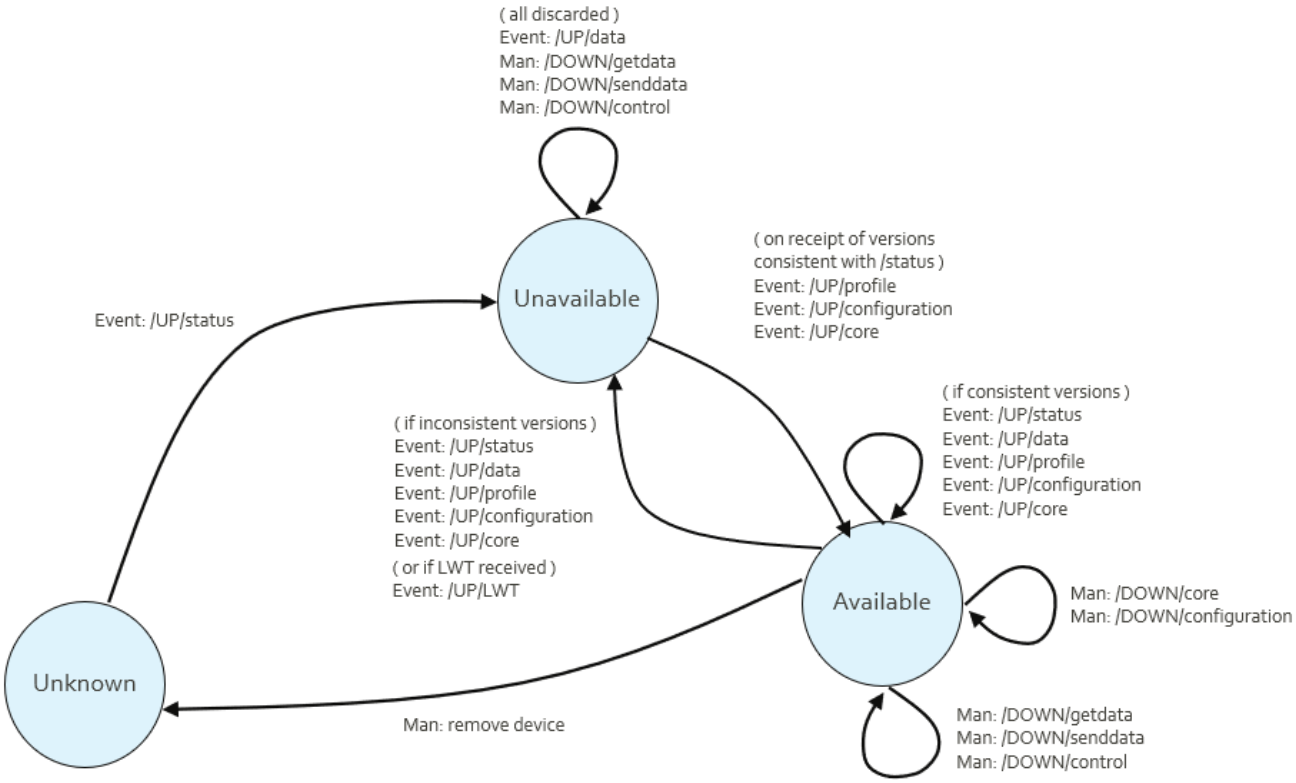


Figure 12: Connection state model

The status message block gives the status information needed for the sections below. This includes the Node ID which can be configured locally to the FD and aids device replacement.

7.1 Status message

The FD sends the status message on connection to the MQTT Broker.

The status message for the protocol uses the Topic name:

[UUID]/UP/status

The QoS is 2, RETAIN is 1.

The status message contains reference information about the Profile and Configuration version. It is sent on every connection, and when the data contained has changed. The status message refers to the following separate elements which are not part of the status message:

- Field Device Profile
  This declares the capability of this range of physical devices. It lists all the available features including the way in which they may be configured. It does not include any configuration. It can be uploaded to the SA, not downloaded. The content of the FD Profile may be a reference to a stored profile rather than the profile itself, in order to save bandwidth, in which case the status message contains that reference and there is no transmitted Device Profile message.
- Core Block(s)
  This can be the device firmware, a device configuration image, or separate application data, and can consist of one or more opaque, usually binary, images. These can be optionally uploaded or downloaded. The term block is used rather than file. An active property indicates the status of the block.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

- Field Device Configuration
  This is a file which carries configuration data and is conformant to the capabilities of the FD Profile. This can be optionally uploaded or downloaded from SA to FD or FD to SA.

The Status message contains:

```
 {
// Field Device Profile Reference (text).
// By convention the format is Manufacturer, Field Device Name,
// Version. Underscores separate the parts of the name, hyphens
// separate version number parts.
// Forward or back slashes or periods are not permitted.
 "pRef":"ACME_RTUOne_1-23",
// Core Block version(s) (text) -- array of names and versions,
// may be an empty array if not present or may be omitted
// altogether. The first Status message from the FD will contain
// this field, if applicable, and subsequent message may omit it
// if unchanged. The names are arbitrary and device-specific.
// If the Required "req" flag is true, then the device is
// requesting that file, either because it is a new device or has
// a fault with its current version.
 "core":[
   {"name":"Firmware",
    "ver":"2.3"},
   {"name":"Program 1",
    "ver":"A",
    "req": true,
    "active":false}
 ],
// Configuration version (numeric) -- may be blank if not present.
// This is the version of the configuration settings data, and is
// managed and incremented by the device. Must match any
// configuration sent from FD to SA.
 "cVer":123,
// Configuration Required from SA (default
// false if not required).
// If some received and processed configuration places the device
// into an error state, this will be true.
 "cReq":true,
// If a configuration message was received and processed
// successfully by the FD and a "source" string was received in
// that message, then the first Status message from the FD will
// contain this source string.
 "source":"62d18075-e5e2-49c2-89e2-abd1cc8025af",
// Time Required from SA (default false if not required).
// Requires the clock set command attribute in the profile.
 "timeReq":true,
// Optional text concerning state of configuration, e.g. message
// explaining problems. Default is an empty string or not
// present.
 "cState":"",
// Optional state of the device -- whether it is actively
// monitoring and available for controls. Default is false.
// A device may report this is true until it receives valid
// configuration.
 "inactiveState":true,
// Node identification -- an optional unique string used for
```

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

```
// device replacement scenarios. I.e. an SA could use this to
// enable replacement of devices with different UUID but
// the same "nid". Must match any configuration sent from
// FD to SA. The SA may use this to determine the desired
// configuration to send, if cReq is true.
 "nid":"123ABC",
// Date and Time this message was created.
 "time":1697613172001
 }
```

A typical example of this message is shown here (with added spacing):

```
 {
   "pRef":"ACME_RTUOne_1-23",
   "core":[
     {"name":"Firmware","ver":"2.3"},
   ],
   "cVer":42,
   "time":1697613172001
 }
```

Core block file version identification text is arbitrary, that is, the protocol does not mandate a version syntax, or that version numbers must increment. The FD specifies through its Status message that files are required, although the SA can independently force updates down to the FD if downloads are supported. Complete core block files are uploaded or downloaded, though not necessarily in a single message.

The FD Configuration version cVer uses ascending version number integers, which, if the SA and FD have different versions and the Required flag is false, the SA determines which to upload or download by reference to the later version number. The FD Configuration can optionally be sent from SA to FD and is mandatory from FD to SA.

7.2 Last Will and Testament

Use of the Last Will and testament (LWT) is optional. The LWT provides a presence indication which is compatible with MQTT.

Each client can optionally specify its LWT, a normal MQTT message with Topic, retained flag, QoS and payload, when connecting to a Broker. The Broker stores the message until it detects that the client has disconnected ungracefully. If the client disconnects abruptly, the Broker sends the message to all subscribed clients on the Topic, which was specified in the LWT.

The LWT for the protocol uses the Topic name:

[UUID]/UP/lwt

The QoS is 2, and the message is:

```
{"state":"Fault", "time":1694773266000}
```

The text of the state is device specific. The time is set by the FD when the LWT is created.

7.3 Field device replacement

FDs may need to be replaced on-site, due to battery changes, upgrades, or hardware failure. To cover this situation, the device Node ID (NID) is transferred from the old device to the new. The new device then begins to communicate with the SA.

The device can be fully configured locally by the on-site staff or may require configuration to be pushed down to the device from the SA. In the latter case the device presents an existing NID in its Status message but coming from a new UUID. The SA can either accept the new device silently or require user

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

input to accept the replacement device as the new FD. The SA may be able to reliably verify identity using cryptographic methods.

SA behaviour in this case is beyond the scope of the communications protocol.

Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

8 Security

The FD Profile has a security property which is a list of the level numbers supported by the device. These numbers are for reference and do not play a part in controlling communications at present. If an FD supports, for example, Levels 0 and 2, then it does not support Level 1. If unspecified the security level is 0.

The Lucid Protocol endeavours to be open and flexible. Because the protocol is used by devices, with different security needs, in applications with varying threat profiles, a choice is offered to users and implementors of the protocol. Depending on the security environment of the field device, one of six levels of security can be used ranging from Level 0: none (or external security mechanisms are already provided) to Level 5: end to end. Levels 1-4 describe uses of TLS from no authentication to full authentication.

Lucid Protocol security levels:

- **Level 0: No encryption**
  Securing communications in this case relies on the inherent security of a communications channel or other outside IT security mechanisms. Multiple SAs are an option because message payloads are unencrypted.
- **Level 1: TLS encryption without Authentication**
  Secures data traffic to the first hop; all traffic is encrypted between FD and MQTT Broker. No certificates are stored on the Broker or FD; device and Broker authentication are not possible. However, communications are secured against sniffing messages in transit. There is no mandated encryption of messages stored on the Broker or during onward transmission. Message encryption at rest and during onward transmission should be secured through additional IT mechanisms. TCP/IP = **TLS 1.2+**, UDP/IP = **DTLS 1.2+**. Multiple SAs are an option because message payloads are unencrypted.
- **Level 2: TLS encryption with Broker Authentication**
  Secures data traffic to the first hop; all traffic is encrypted between FD and MQTT Broker. The TLS/DTLS Certificate of the chosen Broker is stored on the FD allowing the FD to authenticate the Broker before sending MQTT traffic. Device authentication is not possible. Communications are secured against sniffing messages in transit. There is no mandated encryption of messages stored on the Broker or during onward transmission. Message encryption at rest and during onward transmission should be secured through additional IT mechanisms. TCP/IP = **TLS 1.2+**, UDP/IP = **DTLS 1.2+**. Multiple SA are an option because message payloads are unencrypted.
- **Level 3: TLS encryption with Field Device Authentication** – Secures data traffic to the first hop; all traffic is encrypted between FD and MQTT Broker. The TLS/DTLS Certificates of all FDs are stored on the Broker allowing the Broker to authenticate FDs during connection establishment; Broker authentication by the FD is not available. Communications are secured against sniffing messages in transit. There is no mandated encryption of messages stored on the Broker or during onward transmission. Message encryption at rest and during onward transmission should be secured through additional IT mechanisms. TCP/IP = **TLS 1.2+**, UDP/IP = **DTLS 1.2+**. Multiple SA are an option because message payloads are unencrypted.
- **Level 4: Fully Authenticated TLS encryption**
  Secures data traffic to the first hop; all traffic is encrypted between FD and MQTT Broker. The TLS/DTLS Certificate of the chosen Broker is stored on the FD allowing the FD to authenticate the Broker before sending MQTT traffic. The certificate of the FD is stored on the Broker allowing the Broker to authenticate the FD during connection establishment. Communications are secured against sniffing messages in transit. There is no mandated encryption of messages stored on the Broker or during onward transmission. Message encryption at rest and during onward transmission should be secured through additional IT mechanisms. TCP/IP = **TLS 1.2+**, UDP/IP = **DTLS 1.2+**. Multiple SA are an option because message payloads are unencrypted.
- **Level 5: Authenticated end-to-end encryption**
  Payload encryption of some or all messages. Message header unencrypted. The FD can authenticate the SA and the SA is able to authenticate the FD. Encrypted communications are only possible

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

with a single SA. Messages are encrypted/decrypted by FDs and SAs, allowing them to pass through otherwise unsecured networks and Brokers. This allows for simpler network setups while providing reassurance of privacy and authenticity.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

9 Class names and ontology

The className field is used to indicate to consumers of FD data that points are of specific physical types. For example, some analogue signals could be flows which would indicate to the SA that specific display characteristics are associated like a flow meter graphic.

Users can enter any text into this field, generally specific to the type(s) of SA they have.

Users can separate parts of the class name text with '**.**' Or '**/**' for example to indicate a structure.

9.1 Fixed class names

The className of some items can use some common definitions listed here. This is to ensure consistency for device management. By allowing the Configuration file to define these items, signals such as Battery can be assigned to different device terminals, which is sometimes the case for identical FDs. The className field is used so that the actual point name can be user-configured.

These classNames are prefixed by '**L.**' to indicate that this document defines them.

These items are optional. FDs do not have to include a Battery status, but the use of these is recommended.

The className field structure is not defined, allowing these items to be structured as the application requires.

Reporting functionality of the points is as specified in the Configuration.

Table 23: Fixed class names

"className" field	**Type**	**Purpose**	**Specifics**
L.FD.Battery	Binary input	Battery status	State 1 is healthy
L.FD.Mains	Binary input	Main power input	State 1 is healthy
L.FD.Signal	Analogue input	Generic common signal indicating communications signal strength	Top of configured range is highest strength
L.FD.IMSI	String	IMSI of current cellular modem	
L.FD.IMEI	String	IMEI of current cellular modem	
L.FD.CCID	String	CCID of current cellular modem	
L.FD.Serial	String	Serial number of the field device	
L.Loc.Latitude	Analogue input	FD location	WGS84 Latitude
L.Loc.Longitude	Analogue input	FD location	WGS84 Longitude
L.Loc.Accuracy	Analogue input	FD Location accuracy	m
L.Loc.Altitude	Analogue input	FD Altitude above sea level	m
L.Loc.Direction	Analogue input	FD Direction	degrees
L.Loc.Speed	Analogue input	FD Speed	m/s
L.FD.Batt.Main.Volt	Analogue Input	"Main" FD battery voltage	Volts
L.FD.Batt.Main.Charge	Analogue Input	Main battery capacity remaining	%
L.FD.Batt.Sec.Volt	Analogue input	Secondary battery voltage	Volts
L.FD.Batt.Sec.Charge	Analogue Input	Secondary battery capacity remaining	%
L.FD.Temp.Int	Analogue Input	FD internal temperature	deg.C
L.FD.Temp.Int.Min	Analogue input	Minimum Internal Temperature	deg.C
L.FD.Temp.Int.Max	Analogue input	Maximum Internal Temperature	deg.C

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

”className” field	Type	Purpose	Specifics
L.FD.Temp.Int.Mean	Analogue input	Mean Internal Temperature	deg.C
L.FD.Temp.Modem	Analogue Input	Modem temperature	deg.C
L.FD.Temp.Ext	Analogue Input	FD external temperature	deg.C
L.FD.Temp.Ext.Min	Analogue input	Minimum external Temperature	deg.C
L.FD.Temp.Ext.Max	Analogue input	Maximum external Temperature	deg.C
L.FD.Temp.Ext.Mean	Analogue input	Mean external Temperature	deg.C
L.FD.Humidity	Analogue input	FD Humidity	%
L.FD.Humidity.Min	Analogue input	FD Maximum Humidity	%
L.FD.Humidity.Max	Analogue input	FD Miminum Humidity	%
L.FD.Humidity.Mean	Analogue input	FD Mean Humidity	%
L.FD.WAN.Tech	String	WAN technology	
L.FD.WAN.MCC	String	Mobile Country Code	0-999
L.FD.WAN.MNC	String	Mobile Network Code	0-999
L.FD.WAN.LAC	String	Location Area Code (or Tracking Area Code)	
L.FD.WAN.TAC	String	Tracking Area Code	
L.FD.WAN.BSIC	String	Base Station Identity Code (2G)	
L.FD.WAN.PSC	String	Primary Scrambling Code (3G)	
L.FD.WAN.Channel	String	ARFCN (2G), UARFCN (3G) or EARFCN (4G)	
L.FD.WAN.Signal	Analogue Input	Signal Strength	dBm

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

10 Field device profile extensions and layout

10.1 Vendor extensions

Vendor-specific properties and objects can be added through the following syntax. The SA Profile specifies whether the SA itself can support such extensions.

10.1.1 New properties

Field Device vendor features can be added in the form of new configuration fields, which allow functionality not covered by this specification to be configured. These must be identified with a deviceSpecific device property. They may be made generic features in future, at which point the property is removed and their names may also change. Examples of these could be Modbus server scanning parameters, loop settle time, or pulse scaling.

10.1.2 New objects

Vendor-specific object types may also be added in the same way, via the specificToProfile object type property.

10.1.3 Translation

A consistent way of representing translatable strings is included for supporting other languages within the Device Profile. At present the English form of the strings is defined by this standard.

10.2 Suggesting SA UI layout

The principal data about each object type are the configFields. These are the configuration data properties and their attributes.

Object Types define the field information and form layout for an object type, for example Field Device, point, and channel. An object type contains information around the properties that it supports, and form layout information that can be used as a starting point for the layout of a configuration form for the object. However, it is up to the SA how this configuration is presented to the user, and array types may require the SA to implement non-form configuration UI.

10.2.1 configFields

This is a JSON schema that describes both the configuration properties of the object type and the JSON format that the SA should output the Configuration data in. Each property must be a simple type, or an array of simple types, and can include simple JSON property validation information such as minimum, maximum etc. Enumerations are also supported. The information also includes a display name as a prompt for the field and a long description which could be used as a tooltip or similar UI element.

10.2.2 Group

A group allows logically related fields to be grouped together, allowing the SA UI to choose whether to show that these fields are related. This element is a single optional text field which has optional dotted names which can infer a hierarchy. For example, group: Actions.State 0 could be a boxed section marked Actions and a collection of items related to State 0 which could be on a single row. There is no direct inference from the group to any specific layout.

There is also an order property which fields may have to indicate how they are to be ordered on the SA UI. This is because the ordering of JSON fields is not guaranteed to be preserved. The order property is an ascending number and may have a decimal value if fields are required to be inserted between existing consecutive fields.

A typical format of an expression to control the visibility of a field is, by example:

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

```
"enable": "controllingfieldname=1"
```

which makes that item visible if the field value of controllingfieldname is equal to 1.

Refer to the JSON Profile and Schema for further details.

Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

11 Example use cases

This section details typical use cases to demonstrate the range of functionality.

NOTE:	**These cases are examples only. The text is formatted for clarity, but there will be no additional non-printing characters in the deployed messages. The [UUID] text would be replaced by the actual Field Device UUID.**

11.1 Simplest case

This case is explained by the protocol messages and content. Note that in this case messages are sent from FD to SA only.

FD sends Status message:

[UUID]/UP/status

```
{ "pRef": "ACME_RTUOne_1-23",
     "cVer": 1,
     "cReq": false,
     "time": 1694695035000
}
```

It contains a reference to the FD Profile, and the SA is understood to have this already, so that it does not need to be sent.

1. FD sends Configuration to the Broker as a retained message on first connection. It could be that the device and points have been pre-configured in the SA, in which case no configuration needs to be sent. If the device is set up or designed to do so, on first contact it sends all configuration to the SA.

[UUID]/UP/configuration

```
{ "cVer":1,
... <etc>
}
```

2. FD now sends data:

[UUID]/UP/data

```
{"p":[{"ai":1,"v":[12],"q":[192],"ts":[123456789321]}]}
```

11.2 A configurable field device

1. This device extends the above by allowing the SA to send configuration to it.

In the FD Profile this line is present:

```
"canConfigDown":true
```

Configuration download:

[UUID]/DOWN/configuration

```
{ "cVer":2,
... <etc>
}
```

3. On receipt of new configuration, or any changed Bulk file, the FD sends a revised Status message back to the SA confirming the version.

[UUID]/UP/status

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

```
{ "pRef": "ACME_RTUOne_1-23",
     "cVer": 2,
     "cReq": false,
     "time": 1694695035000
}
```

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

12 Supervisory application profile document

This document currently plays no part in the protocol. It lists the features and characteristics of the SA to enable a user to determine how the SA interoperates with the FD.

Vendors of a Lucid SA must issue this document to the WITS PSA and to users.

The SA Profile document is in a JSON format.

```
{
 // The Lucid version number to which this profile
 // document relates. If more than one Lucid version is
 // supported then multiple documents would be issues,
 // each specifying those versions would be applicable to
 // the SA.
  "lucid": 3,
 // Identification of the Supervisory Application.
  "SAName": "ACME Lucid",
 // Version of the Supervisory Application.
  "SAVersion": "1.1",
 // Number of decimal places the SA will store and process time
 // stamps.
  "timePrecision": 3,
 // Capability for file data transfers in either direction,
 // denotes whether each block can be uploaded or downloaded by
 // the SA Application. (Every SA must be able to receive
 // the configuration).
 // If true, then the user needs to be able to enable or
 // disable this functionality.
  "canCoreUp": false,
  "canCoreDown": true,
  "canConfigDown": true,
 // Capability for sending "getdata" requests.
  "canGetData": true,
 // Capability for sending "control" requests.
  "canControl": true,
 // Capability for sending "override" requests.
  "canOverride": true,
 // Capability for time setting using the protocol.
 // If true, then the user needs to be able to enable or
 // disable this functionality
  "canSetClock": true,
 // Compression:
  "supportCompression": ["gzip"],
 // Capability for supporting MIME points and data. The MIME
 // object will be stored by the Supervisory Application. There
 // is no obligation for any specific processing or display.
  "supportMIME": true,
 // The maximum number of limits per analogue point.
  "maxAnalogueLimits": 8,
 // The maximum number of limits per counter point.
  "maxCounterLimits": 4,
 // Capability of supporting new vendor extension properties.
  "supportVendorProperties": false,
 // Capability of supporting new vendor extension objects.
  "supportVendorObjects": false,
 // Object types supported by this SA (all are assumed if this
 // is not present).
```

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

```
  "supportedObjectTypes": ["device", "channel", "connection",
  "connectionSchedule", "deviceLink", "point", "binary",
  "analogue", "counter", "string", "analogueLimit",
  "counterLimit", "noChange",
  "extPoint", "limitProfile", "controlProfile", "incidentLog"],
 // Calculated points are a feature of point objects,
 // but an SA may or may not support configuring them.
  "supportCalculatedPointConfig": false,

}
```

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

A Appendix: JSON files

The following files in the code repository provide further information. See https://github.com/LucidProtocol/protocol-json.

- **lucid.schema.json** Provides information on the Lucid schema.
- **Profile.json** Provides an example Core Field Device Profile.
- **Configuration.json** Provides an example FD Configuration.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

B Appendix: Topic summary

This section lists the topics used by the protocol.

Table 24: Topics used by the protocol

Topic Name	Note	QoS	Retained
[UUID]/UP/lwt	Last Will and Testament	2	1
[UUID]/UP/status	Status Message	2	1
[UUID]/UP/profile	FD Profile	1	1
[UUID]/DOWN/core/block/<name>/<version-text>	Data Block	0	0
[UUID]/DOWN/core/request	Request Data Block	0	0
[UUID]/UP/core/block/<name>	Data Block	0	0
[UUID]/DOWN/configuration	Configuration	1	0
[UUID]/UP/configuration	Configuration	1	1
[UUID]/DOWN/getdeviceinfo	Configuration	0	0
[UUID]/UP/data	Data	1	0
[UUID]/DOWN/getdata	Data	0	0
[UUID]/DOWN/setclock	Clock	0	0
[UUID]/UP/getclock	Clock	0	0
[UUID]/DOWN/control	Control	2	0
[UUID]/DOWN/override	Override	2	0

The topics are focussed on verbs, or actions, rather than a data addressing scheme.

The QoS for some messages is variable:

- The SA can, optionally, configure the QoS to be used for data, control, and override messages.
- For data, the QoS is specified in the Configuration.
- For control and override, the SA can specify the QoS to be used.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

C Appendix: Calculated point code

The value of a calculated point is based on the values of one or more other points in the FD. The value is calculated using an expression that contains one or more operations. Operations are encoded using a code array defined in this section. An operation may have zero or more parameters. The number of parameters is fixed for each operation.

An expression and its constituent operations are encoded in Polish Notation (PN). Calculation of the expression may use a Last In First Out (LIFO) stack but all implementation details are specific to the vendor. The description specifies the coding of the expression and any rules significant to the calculation. An example of the encoding and a possible calculation scheme is given in Table 26.

The expression for the calculated point consists of a list of numeric codes and parameters representing the expression and inputs using PN/stack notation. The bytecodes are given in Table 25, showing the operation code, stack before, stack after, and a description of the operation. Each instruction has up to four values:

- The operation code.
- Zero to three parameters needed by that instruction.

Each instruction expects up to three values to be on the stack prior to execution. After execution they are replaced with a single result value on the stack.

C.1 Evaluation of a calculated point value

The calculation of a point value commences when determined by one of the execution strategies that apply to the calculated point.

After calculation, a single result value is present on the stack. An expression is not valid if this is not the case.

For binary inputs and outputs, the value 0 or 1 is used to represent the value.

Where logical operation results or single bit binary values are used, the 0 or 1 is encoded as an integer literal with value 0 or 1, respectively.

For double- or three-bit binary inputs, the values 0 to 7 are encoded as an integer literal.

Table 25 suggests a simple format for expression of the calculations in the Example Syntax column. However, the format of the calculation language may be specific to the configuration application creating the configuration message.

For example, the calculation may be specified using point names or may use a syntax created from the point type and point address. It is the calculation code which is common. This document defines the operation codes to be used irrespective of any language used to express the calculation to a user.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

C.2 Operation codes

Calculations refer to other points by an enumeration of the point type: (ai=0, bi=1, ao=2, bo=3, c=4, s=5) and their point number.

Table 25: Operation codes

Operation Code	Example Syntax	Type	Parameter 1	Parameter 2	Parameter 3	3rd down on stack	2nd down on stack	Top of stack	Result on stack	Description
0	-1337	Integer Literal	Integer value						Number	Integer literal left on top of the stack
1	3.14	Floating point literal	Floating point value						Number	Floating point literal left on top of the stack
10	“AI1”	Source point value	Point type number	Point Number					Number	Point value is read and placed on top of the stack. If the point has never been read, 0 is used
11	Q(“AI1”,3)	Source point quality flag	Point type number	Point Number	Bit number within quality flags				Number	Point quality flag value is read and placed on top of the stack as 0 or 1.
12	T(“AI1”)	Source point timestamp (seconds)	Point type number	Point Number					Number	Epoch time (**seconds** since 1/1/1970) of last point update is placed on top of the stack. If never processed, 0 is used
12	TMillis(“AI1”)	Source point timestamp (milliseconds)	Point type number	Point Number					Number	Epoch time (**milliseconds** since 1/1/1970) of last point update is placed on top of the stack. If never processed, 0 is used
20	-	Unary minus						Number	- Number	Negate number on top of stack
30	+	Add					Number1	Number2	Number1 + Number2	Add two numbers and place on top of stack
31	-	Subtract					Number1	Number2	Number1 - Number2	
32	*	Multiply					Number1	Number2	Number1 * Number2	
33	/	Divide					Number1	Number2	Number1 / Number2	
34	**	Power					Number1	Number2	Number1 ** Number2	

 Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

Operation Code	Example Syntax	Type	Parameter 1	Parameter 2	Parameter 3	3rd down on stack	2nd down on stack	Top of stack	Result on stack	Description
35	MOD	Modulo					Number1	Number2	Number1 MOD Number2	Remainder after division
40	NOT	Logical NOT						Number	Not Number	All logical operators treat zero as false and non-zero as true. The NOT value of a number is 0 or 1
50	OR	Logical					Number1	Number2	Number1 OR Num-ber2	
51	AND	Logical					Number1	Number2	Number1 AND Num-ber2	
52	XOR	Logical					Number1	Number2	Number1 XOR Num-ber2	
53	XNOR	Logical					Number1	Number2	Number1 XNOR Num-ber2	
60	IIF	Immediate If				Condition	Number1	Number2	Number	If Condition is true, result is Number1, else Number2
70	ABS	Absolute Value						Number1	Number	If Number1 is negative, return -Number1
71	INT	Integer						Number1	Number	The integer version of Number (truncation)
80	MAX	Return largest of two numbers					Number1	Number2	Number	The result is the largest of the two parameters
81	MIN	Return smallest of two numbers					Number1	Number2	Number	The result is the smallest of two parameters
90	=	Equal to					Number1	Number2	Number	Result is Boolean evaluation of Number1 = Number2 (false is zero, true is one)
91	!=	Not equal to					Number1	Number2	Number	Result is Boolean evaluation of Number1 != Number2 (false is zero, true is one)
92	>	Greater than					Number1	Number2	Number	Result is Boolean evaluation of Number1 > Number2 (false is zero, true is one)
93	>=	Greater than or equal to					Number1	Number2	Number	Result is Boolean evaluation of Number1 >= Number2 (false is zero, true is one)

 Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

Operation Code	Example Syntax	Type	Parameter 1	Parameter 2	Parameter 3	3rd down on stack	2nd down on stack	Top of stack	Result on stack	Description
94	<	Less than					Number1	Number2	Number	Result is Boolean evaluation of Number1 < Number2 (false is zero, true is one)
95	<=	Less than or equal to					Number1	Number2	Number	Result is Boolean evaluation of Number1 <= Number2 (false is zero, true is one)
100	TimeWeekDay(z)	Day of week						Number1 (epoch seconds)	Number (day of week)	Get the day of week from the Epoch 1-7 (Mon-Sun)
101	TimeHour(z)							Number1 (epoch seconds)	Number	0-23
102	TimeMin(z),							Number1 (epoch seconds)	Number	0-59
103	TimeSec(z),							Number1 (epoch seconds)	Number	0-59
104	TimeDay(z)							Number1 (epoch seconds)	Number	1-31
105	TimeMonth(z)							Number1 (epoch seconds)	Number	1-12
106	TimeYear(z)							Number1 (epoch Seconds)	Number	Year e.g. 2022
107	TimeEpoch								Number	Count of seconds since Jan 01 1970 UTC
108	TimeEpochMillis								Number	Count of milliseconds since Jan 01 1970 UTC
120	Previous()								Number	Get the value of this calculated point which is about to be overwritten

NOTE:	All unused bytecodes are reserved for future Lucid use.

The operation codes do not support any bitwise operators.

The operation codes do not support parentheses as the Polish Notation takes care of this.

There is no support for local time. All times are based on UTC as defined in Lucid.

 Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

C.3 Example

This section presents an example of a calculation, how it is encoded and how it could be calculated.

NOTE:	**The method of calculation is vendor dependent and need not follow the example shown here.**

The example calculation expression takes the average of two analogue points (AI1 and AI2) using the calculation below:

Calculated Point Value = (AI1 + AI2) / 2

This expression is encoded as follows:

Table 26: Example of a calculation

Instruction	Bytecode	Syntax	Type	Parameter 1	Parameter 2
1	33	/	Divide		
2	30	+	Add		
3	10	AI1	Source Point Value (SPV)	0 (Analogue In)	1 (Point number)
4	10	AI2	Source Point Value (SPV)	0 (Analogue In)	2 (Point number)
5	0	2	Integer Literal (IL)	2	

This appears in the configuration as an array of the numbers below (spacing for clarity only):

33, 30, 10, 0, 1, 10, 0, 2, 0,2

C.4 Quality flags

All calculated points have quality flags as any other Lucid point. The rules for calculation of those flags are given below:

- If a calculated point is Offline or its Restart flag is set, then no calculation need be performed for that point.
- Offline for a calculated point is the logical OR of all Offline flags for every point in the calculation together with the inverted On Scan setting of the calculated point. Therefore, if any of the contributing points are Offline/Not On Scan then the calculated point is Offline, also, if the calculated point is set to Off Scan, then it is Offline.
- Restart for a calculated point is the logical OR of all Restart flags for every point in the calculation. If any point in the calculation has not had Restart reset, then Restart is set for the calculated point and hence the value should be disregarded.
- Comms Lost, Remote Forced and Local Forced for a calculated point is the logical OR of the same flags for every point in the calculation. It is not possible to set these for any other reason for the calculated point, they are dependent entirely on the source points.
- Calculated points are either a binary or analogue point, hence the following rules are used.
- For a binary calculated point Chatter is the logical OR of the same flag for all binary points used in the calculation.
- For an analogue calculated point Overrange is the logical OR of the same flag in all analogue points used in the calculation, OR'd with the overrange indication resulting from the calculation. So, for instance, if a calculation multiplies two source numbers together and gets an answer out of range then the Overrange flag would be set irrespective of the state of the source points.
- For an analogue calculated point Reference Error is set to the logical OR of the same flag for all analogue points used in the calculation.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

- Rollover and Discontinuity are never used.
- The State flag for a calculated binary point is the calculated value of the point.
- At start-up and before any calculation is performed, the default value of all flags is 0, apart from Restart which is set to 1.
- Where a calculation uses source points which have a different point type to the calculated point, the flag of the calculated point is determined only from those flags in the source points which match those in the calculated point. Where appropriate, it is recommended that calculations separately deal with non-matching object flags.

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.

D Appendix: Revision history

Table 27: Lucid Protocol Revision History

Version	Date	Description
1	08/10/2018	Initial version
2	17/05/2024	Update version after multiple plugfests
3	23/10/2024	Various corrections and an update to tag type references: • Ticket #51: **Data type enhancements**. Types (ai, bi, ao, bo, c etc.) are now 1st class entities and not attributes of a point. For example: • Ticket #45: New **'WhenUnableToRead'** action on a point • Ticket #44: all times are now consistently in **milliseconds from Epoch** • Ticket #35 / #18: **corrections on data types** for various message fields • Various minor protocol document enhancements and corrections
4	11/05/2026	Minor corrections: • Ticket #83: Corrected Figure 1 to bring in line with current Lucid specification. • Ticket #85: Grammer corrections in section 4.1 • Ticket #90: Use "State Change" reason code for "whenXXX" events. • Ticket #94: Clarified meaning of Repeat Count in section 2.1.4. • Ticket #96: Update details in section 2.1.3.4 to indicate that a full download is required to delete an item.

Also refer to history page at: https://github.com/LucidProtocol/Lucid-Specification/releases

{lucid} Copyright © 2026 WITS Protocol Standards Association. All rights reserved.