CASAMBI

Casambi Data Logging and API

v1, 22.02.2021

Subject of change

www.casambi.com · info@casambi.com

Revision History

Date	Version	Description
22.02.2021	1.0	Created based on Casambi Data Logging document as addition to API documentation.

INDEX

Background

Casambi introduces data logging facilities for end-users, managers of lighting systems and system integrators. Lighting networks can be connected to the Casambi back-end services on the Internet with the gateway function of mobile Casambi apps or equivalent.

The functions of the gateway connections are:

- Storage of network configuration and access credentials for shared access
- Remote access to the network over the Internet with full-featured control and configuration functions
- Maintenance of backup snapshots of network configuration and management
- Delivery of diagnostics information for technical support and troubleshooting with Casambi
- Delivery of network status, activity of individual Casambi-enabled device, devices' control state, sensor readings and detailed OEM details of attached devices for logging purposes

When a network is connected to the Casambi service, it becomes observable by the server-side logging facilities and remotely connected Casambi apps or equivalent integrated systems. Different network and device data that is logged on the server side can also be accessed via Casambi Public API services.

Data and API Services

Data logging contains information about device, device specific details, information about its state and changes in the state. In addition, specific devices can provide more custom details that depends on the underlying platform where Casambi module is integrated. For example, attached DALI device can be source of ODM/OEM details, serial numbers, GTINs, types and special hardware capability information.

By default, API provides “basic” information about network, its devices (and their state), scenes, groups and their structure, including related image/icon data and interface related information. Most of this basic information can be accessed via Casambi REST API requests. Exception to this is the state information of devices that is provided via “unitChanged” events of Casambi WebSocket service.

In addition to basic information, it is also possible to collect more specific datapoint data and device details information via Casambi Public API when this functionality has been enabled from the related Gateway device. This information includes sensor and vendor data layer “Sensor data” and device specific detail rich data layer “Usage of Devices”.

This more specific information (when enabled) can be accessed via Casambi REST API datapoints requests and on some part also from “details” dictionary of “unitChanged” events of Casambi WebSocket service.

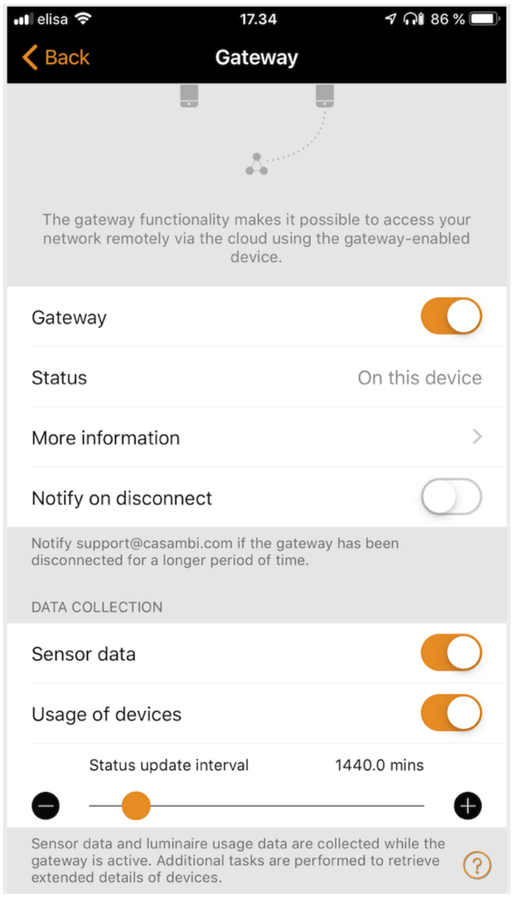
Enabling Data Logging

Logging is enabled for connected networks – ones that have continuous connection to the Casambi back-end services by activation of the Gateway function in network settings in the Casambi app on the gateway device.

Gateway device provides remote access to several local Casambi networks. The number is limited by capabilities of the hardware and software platform to maintain concurrent Bluetooth connections, one for each network.

Scope of logging is controlled with two gateway parameters in the *Data collection* section:

- Sensor data
- Usage of devices



elisa 17.34 86 %

Back Gateway

The gateway functionality makes it possible to access your network remotely via the cloud using the gateway-enabled device.

Gateway: on

Status: On this device

More information

Notify on disconnect: off

Notify support@casambi.com if the gateway has been disconnected for a longer period of time.

DATA COLLECTION

Sensor data: on

Usage of devices: on

Status update interval 1440.0 mins

Sensor data and luminaire usage data are collected while the gateway is active. Additional tasks are performed to retrieve extended details of devices.

"Sensor data" controls logging of:

- Vendor sensors (only those that are specified with the
  "On cloud?" option in the fixture profile)
- Common ("standard") sensors used for Casambi control and supported in the Casambi apps: daylight, presence, temperature, battery level; also, indicators of special abnormal device states (overheated, current overload) are considered sensor data and included in this set.

This sensor data is stored for 60 days and during that time can be accessed via REST API datapoints requests.

"Usage of devices" activates the rest of logging options and additional operations by the gateway to actively collect, aggregate and store data required for periodic logging:

- Status of the device in the network
- State of device controls, i.e. all of the "light control" parameters like dimmers, CCT level, color settings, custom elements, etc.
- Energy counters
- Manufacture details of the controlled DALI gear
- Extended diagnostics and maintenance data that can be retrieved from controlled (DALI) devices

This kind of device specific data can be partly accessed via REST API datapoints requests (within 60 days from its creation) and also in "real-time" via Websocket "unitChanged" event details.

Note that the "unitChanged" event data is delivered (via WebSocket) whenever device state change occurs, and it is only partly stored to device state cache on the server side.

Casambi "Datapoints" REST API

Collected data stored temporarily in the Casambi Cloud service where it can be consumed by third-party systems that perform analytical, monitoring (and later maybe also control) functions.

The following sections describe what data is stored, under what commonly defined data type name ("dataType" filter in REST API filter options) and measurement units.

Refer to the API developer reference for details of the REST API calls to retrieve the historical records of the collected data: `https://developer.casambi.com/#request-network-datapoints`

Summary of Data Types

Following chapters present each data type filter used in REST API datapoints request (stored for 60 days). These filter options are sensors, vendor, control, energy, details, status.

Note that all properties presented here might not be available via datapoints REST API requests nor via WebSocket events yet.

Sensors

Sensors are logged as is – numeric value (for INTEGER or FLOAT types) with names defined in the fixture configuration ("Logged as" attribute).

Following table presents (datapoint) sensor data keys and values. Note that presence values include also information about the duration of present/absent state.

Data Property and Logging Details	Type, Units, Range	Contents
Lux max. One change per minute with minimum change of 8	NUM (lux)	Illuminance in lux units 0 ~ not available
presence	NUM (sec)	Presence event: activation or deactivation of the sensor N > 0: duration of the active state (seconds) N < 0: duration of the inactive state (seconds)
daylightcontrol max. one change per minute	INT [0..254]	[0..254] level for application on daylight-control scenes in Casambi network. This type of sensor-like data is exported for DALI sensors (or equivalent controller-source) exerting direct control of dimming level for other targets (devices, scenes, groups or whole network).
battery_level max. one change per minute	P	[0..1] represents the percent of battery level
overheat	BOOL	Built-in overheat-indicator in some Casambi devices (CBU-TED) or equivalent indicator-input in custom designs.
overcurrent	BOOL	Built-in overheat-indicator in some Casambi devices or equivalent sensing input in custom designs.

Vendor sensors (vendor)

Vendor sensors are custom data provided by device vendor and are logged as is – numeric value (for INTEGER or FLOAT types) with names defined in the fixture configuration (“Logged as” attribute).

Vendor sensor details are fixture specific, defined in the Casambi module configuration and provided from the host system via the Casambi Firmware Extension Interface.

Note that vendor sensors data may provide values equivalent to some of the Casambi-defined standard logged data – sensors, device properties, energy counters, etc., that might or might not follow the standard names used in this specification.

Control state (control)

List of (datapoint) control properties and their values.

Data property	Type	Contents
dimmer **dimmer[2..8]** *)	P	Intensity 0-100% (as 0-1 normalized value) for corresponding dimming control input. Up to 4 dimmers are supported currently (up to 8 in the fixture)
slider **slider[2..8]** *)	NUM	Value of custom slider in the [Min..Max] range as defined in the fixture profile.
cct	NUM	Color temperature in Kelvins, rounded to nearest 10K value.
vertical	P	Position of the ratio-slider that redistributes single control-dimmer between two output channels.
white	P	0..100% of the "White" control channel
colorsource	INT	Active type of output, typically present in the fixture modes where color(RGB or XY) and TW outputs are exclusive: **0:** tunable-white(CCT based) output **1:** RGB output **2**: XY output
colorLevel * *)	P	"White/Color" slider value that adjust intensity (multiplies the current dimming-level) of color and White/TW channels. colorLevel = The normalized color level in range [0..1]. White/TW is at 100%; color grows 0..100%
whiteLevel * *)	P	"White/Color" slider value that adjust intensity (multiplies the current dimming-level) of color and White/TW channels. whiteLevel = The normalized white level in range [0..1]. White/TW declines [100..0%], colors at 100%
onoff *)	BOOL	State of the custom On/Off toggle control.
button *)	BOOL	State of the custom Button control (pulsed input)
pushbutton *)	NUM (sec)	State of the custom PushButton control Duration in the pressed state is reported.

*) index [2..N] is added to the name for multiple instances of the control, e.g. "onoff2"

**) The irregularity is due to internal resolution of the control channel (6-bit) At the level 31 (~ 0.492 if normalized), both white and color outputs are 100% (of current dimming level).

Note that default logging policy for control is “at most once in a minute” if not specified otherwise.
When the lighting control happens – for manual or automated actions – values of the control parameters transit through series of changes. Relevant “key point” and viably timed snapshots of the control parameters are recorded for a detailed history of all control activities in the network.

Energy data (energy)

Energy counters are device specific information that can be received from DALI devices.

Energy counters and dynamic electrical properties of devices are represented here:

- Total energy
- Energy consumed (resettable) since the last reset
- Power (W or VA) with several specific subtypes: active/apparent/load-side
- Current (A) for main-input or LED-output connections
- For multi-driver assemblies or multi-device setup, e.g. controlled DALI network, individual “sub-devices” behind a Casambi module may provide own statistics and dynamic data. Total energy and active power can also be aggregated at network level, providing single value suitable for monitoring and management purposes.

Common (datapoint) energy details are presented in the table below.

Data property {Type}	Unit	Contents
`energy_total`	Wh	Total energy consumed by the device. DALI DT51 *ActiveEnergy*
`energy_rst`	Wh	Total energy (resettable). NOTE: it is not a standard DALI DT51 property
`output_power`	W/VA	DALI DT51 *ActivePower*
`energy_ts {TS}`		Timestamp (Unix time) of the acquiring the set of related values. This property may not be logged as a separate value if incorporated into timestamps of each of the data-point records.

All changes are logged with 60s restriction interval, but practically, the implemented procedures for automatic updating of the energy counters run at much large intervals (defined in the application settings).

Supported devices have device type “51”, “DEXAL” or “SR”.

Energy counters are read periodically from all addressable DALI drivers in the Casambi network when "Usage of devices" option is enabled in the gateway settings. Each scan cycle updates the timestamp property after the full set of energy counters is retrieved via the Casambi module.

Details

General device properties/details

Data property	Type	Contents
address	STR	Logical address of device. For DALI devices short addressed are "0", "1", … "63"
serial	STR	Serial identification number It is expected to be unique for given GTIN or at least for available combination of OEM/model identifiers.
GTIN	STR	GTIN identifier DALI: MB #0 GTIN
model	STR	Name of the device model Note that value is limited by 24 bytes currently and may be shorter than the DALI MB#1 contents may provide (up to 40 bytes)
ODM	STR	*Original Device Manufacturer* name for the device. The company names are currently derived from known "GS1 prefixes" in GTIN value – and presented as short names like "OSRAM", "Signify", "ERCO", "Tridonic", "Helvar", etc. for informative purposes, for possible augmentation of the device model name with the ODM name.
device_type	STR	Combination of handled device-type IDs with the ":" separator, for instance: "6:8", "8:50:51", "SR:6", "DEXAL:6:51:52" (6 ~ DT6; 8 ~ DT8; 51 ~ DT51, etc.) Also, other supported vendor specific extension can be included: "DEXAL" ~ OSRAM DEXAL "SR" ~ Philips SR Note that full scanning of all supported device types and feature extensions is not performed, and only relevant DT identifiers are exported.
manufacture_date	STR	String "*YYYY:WW*" (year and week) or *YYYY-MM* or similar date format. DALI: MB #1 Manufacture Date (DT50)

Diagnostic and maintenance details

Data property	Unit	Contents
system_starts		Counts the number of device starts as a result of power cycle of the external supply. DALI DT52 *ControlGearStartCounter*
operating_time	s	Counts the control gear operating time in seconds while the device is powered regardless of the status of attached lamp. DALI DT52 *ControlGearOperatingTime*
input_voltage	V	RMS value of external supply voltage: **Mains Voltage** (AC) or **Power Supply Voltage** (DC) DALI DT52 *ControlGearExternalSupplyVoltage*
input_current	A	**Mains Current** (AC) or **Consumed Current** (DC)
output_voltage	V	Actual output voltage of the control gear. DALI DT52: *LightSourceVoltage*
output_current	A	Actual control gear output current DALI DT52: *LightSourceCurrent*
output_current_p	% [0..1]	Output current in % related to the nominal output current settings of the control gear. DALI DT52: *ControlGearOutputCurrentPercent*
output_power_p	% [0..1]	Current power factor of the device DALI DT52 *ControlGearPowerFactor*
temperature_max	C°	Maximum registered temperature by the monitoring system. DEXAL: *Case Temperature (Max)*
temperature	C°	Current operating temperature of the device. DALI DT52: ControlGearTemperature [-60 .. +193] C° DEXAL: *Case Temperature*
failure_status	bits	The integer value provided bitfield that corresponds to the DALI DT52: **ControlGearFailure** flags

The default logging policy `"maximum one change in a minute"` applies to all numeric properties.

If there are more frequent changes, the most recent within 1 minute from the last recorded change will be added as the next sample. This creates reduced historical view of the varying data series.

Table 1 - ControlGearFailure condition flags.

Index of failure condition flag	**Name of failure condition flag**	**Name of related failure condition counter**
0	ControlGearOverallFailureCondition	ControlGearOverallFailureConditionCounter
1	ControlGearExternalSupplyUnderVoltage	ControlGearExternalSupplyUnderVoltageCounter
2	ControlGearExternalSupplyOverVoltage	ControlGearExternalSupplyOverVoltageCounter
3	ControlGearOutputPowerLimitation	ControlGearOutputPowerLimitationCounter
4	ControlGearThermalDerating	ControlGearThermalDeratingCounter
5	ControlGearThermalShutdown	ControlGearThermalShutdownCounter

Light-source details

Data Property	Unit	Contents
`lamp_time`	`s`	Counts the light source operating time in seconds. DALI DT52: *LightSourceOnTime*
`lamp_starts`		Counts the starts of the light source. DALI DT52: *LightSourceStartCounter*
`lamp_temperature`	`C°`	Actual operating temperature of the device. DALI DT52 *LightSourceTemperature*
`lamp_fail_status`	`bits`	The integer value corresponds to the bitfield [4:0] DALI DT52 *LampFailure* condition flags.

Table 2 - LampFailure condition flags.

Index of failure condition flag	**Name of failure condition flag**	**Name of related failure condition counter**
0	LightSourceOverallFailureCondition	LightSourceOverallFailureConditionCounter
1	LightSourceOpenCircuit	LightSourceOpenCircuitCounter
2	LightSourceShortCircuit	LightSourceShortCircuitCounter
3	LightSourceThermalDerating	LightSourceThermalDeratingCounter
4	LightSourceThermalShutdown	LightSourceThermalShutdownCounter

Status

Status properties of a Casambi device in the network.

Data Property	Type	Contents
`online` time resolution of 15 seconds	`BOOL`	Observation of the online status of the unit in the mesh-network. Units may go offline for a number of reasons – from being powered down or undergoing smart-switching action, to resets as result of firmware update or poor communication link with the part of the network where observer (mobile gateway) resides.
`scene`	`INT`	Active scene ID on the unit [0 = no scene] Scene identifiers provide reference to full scene details that can be retrieved from network configuration data.

`priority`	`INT`
	
	`1..15`
		Priority of the effective control (and corresponding output state) on the unit. Smaller ID have higher priority in the Casambi control-stack handling:
		`0: “undefined”`
		`1: “emergency”`
		Emergency
		`3: “manual”`
		Manual control (application, push-buttons, switches)
		`5: “event-priority-date”`
		Date-based scheduled events that override presence
		`6: “event-priority-weekday”`
		High-priority scheduled event that override presence
		`8: “presence”`
		Presence action
		`11: “event-date”`
		Date-based scheduled events (Timers / Date);
		also higher-priority weekday timers use this level
		`12: “event-weekday”`
		Scheduled events (Timers / Weekdays)
		`15: “startup”`
		Start-up animation
		
		More intermediate priority levels may be introduced later.

`condition` time resolution of 5 seconds	`STR`	Overall condition status for reporting abnormal device states. It is based on the **unit condition** code shown in the diagnostics views of Casambi apps and the Casambi Utility. Some of the conditions are defined after the DALI specification and correspond to reports from controlled DALI devices. `Values:`

ok	Nominal state
n/a	Not available
hw_failure	Generic hardware failure reported
overload	Generic current overload indicated, protection mechanism may be active on the unit
overheated	Exceeding normal thermal range; thermal protection of the unit may be activated DALI: thermal derating report
thermal_overload	DALI: thermal overload report
lamp_failure	DALI: ballast failure report
driver_failure	DALI: gear failure report
hw_not_found	DALI: gear does not respond
short_circuit	Light-source short circuit (DALI)
open_circuit	Light-source open circuit (DALI)
io_error	Signaling problem (DALI bus, EXTIF)
configuration_failed	Internal Casambi error; DALI process fails at start-up (does not complete normally)

Casambi WebSocket Service

Casambi WebSocket service offers “real-time” event data via “unitChanged” message. In addition WebSocket service delivers “networkUpdated” and “peerChanged” messages whenever network has changed or WebSocket client has joined or left the current WebSocket “wire".

Let’s have a closer look at “unitChanged” event as it offers lot of device specific information.

UnitChanged Event

These “unitChanged” messages contain basic device info and event (and device) specific “details” data. Specially DALI devices offer more details data within event messages.

More information about “unitChanged” events basic device info can be found from Casambi Public API WebSocket documentation: https://developer.casambi.com/#ws-event-message

In this documentation we are going to focus on the “controls”, “sensors” and “details” properties of the “unitChanged” event and their possible values.

Note that this kind of event data is not stored to the database on the server side and is only partially kept in server’s cache of last-known device states. To store event for longer time it is recommended to save it (as it occurs on WebSocket) on the client-side storage.

Following example shows message content of “unitChanged” event (of related DALI device) received via WebSocket when state of device has changed in the network.

```
{
  "controls": [
    {
      "type": "Dimmer",
      "value": 0.5
    }, {
      "source": "XY",
      "type": "Colorsource"
    }, {
      "type": "Color"
      "x":0.368343917,
      "y":0.13,
      "rgb":"rgb(255, 143, 255)"
    }, {
      "min": 2700,
      "max": 6000,
      "type": "CCT",
      "value": 6000,
      "level": 1
    }
  ],
  "method": "unitChanged",
  "priority": 15,
  "id": 45,
  "groupId": 0,
  "position": 2,
  "address": "df89c844c301",
  "name": "Example Luminaire",
  "fixtureId": 1234,
  "type": "Luminaire",
  "condition": 0,
  "wire": 10,
  "sensors": {},
  "online": true,
  "activeSceneId": 0,
  "dimLevel": 0.5,
  "details": {
    "DALI": {
      "SERIAL.4456935198821319392": {
        "GTIN": "8718696698181",
        "ODM": "Philips",
        "address": "0",
        "device_type": "6:SR",
        "energy_resettable": 900,
        "energy_total": 1.5,
        "energy_ts": 1560345734.924028,
        "lamp_time": 79,
        "model": "Xitanium 60W 0.08-0.35A 300V SR 230V",
        "operating_time": 16685,
        "output_power": 300,
        "serial": "4456935198821319392,
        "status": "02",
        "system_starts": 45
      }
    },
    "OEM": "Casambi",
    "fixture_model": "ExampleXY"
  },
  "status": "ok",
}
```

Note that the properties "activeSceneId", "dimLevel" and “status” are additional key values presenting the overall dimming level, ID of active scene if any and string format presentation of device status. If there are more than one dimmer in the device, then “dimLevel” will be counted as average of all dimmer values.

Controls

The “controls” property provides similar information as shown in the datapoints REST API chapter earlier but in this case the data is converted to more usable format with additional control’s parameters included and having more than just “value” in some cases.

Controls data of the “unitChanged” event is an array that contains “control” items. Order of items corresponds to definition of control fields in devices’ fixture profile. For example, when there are multiple dimmers or custom elements, they appear in the same order as in application UI and in the “controls” descriptions in fixture data (see example event above).

Each “control” item of this “controls” array is an object that has at least has key property “type” and in most cases also a key property “value”.

List of control item types and their values.

Control Type	Value Type	Related Value(s)
`Vertical` **`White`** **`Dimmer`**	`P`	Vertical/White/Dimmer value given as: value = Intensity 0-100% (as 0..1 normalized value) for corresponding control input.
`Slider`	`NUM`	Slider control value given as: value = Position of custom slider in the [Min..Max] range as defined in the fixture profile. label = Label name for the custom slider (if given).
`CCT`	`NUM`	Color temperature values given as: value = Color temperature in Kelvins, rounded to nearest 10K value. min = Minimum temperature value in Kelvins. max = Maximum temperature value in Kelvins. level = Color temperature normalized to range [0..1].
`Temperature`	`NUM`	Ambient temperature value given as: value = Temperature value in degrees (C)

Color	NUM	Color based on Colorsource (Hue/Sat or XY). In either case will include "rgb" presentation of the color. In case of XY the "rgb" value is more experimental -> not very accurate. rgb = RGB coded version of the color like "rgb(255, 0, 0)". hue = Hue value of the Hue/Sat color. sat = Saturation value of the Hue/Sat color OR.. x = X value of the XY based color (more experimental). y = Y value of the XY based color (more experimental).
BatteryLevel	P	Battery level of device given as: value = Value between 0-100 (%).
Overheat	STR	Overheating status of device given as: status = One of "ok", "overheated" or "cooling".
Lux	NUM	Lux value given as: value = Current Lux value.
DaylightControl		DaylightControl value given as: value = Current Lux value. level = Current Lux value normalized to range [0..1].
Presence	STR	Current state of presense sensor given as: status = One of "absent", "present", or "lingering".
Colorsource	INT	Active type of output, typically present in the fixture modes where color(RGB or XY) and TW outputs are exclusive: source = One of "RGB", "XY" or "TW".
ColorBalance	P	"White/Color" slider values that adjust intensity (multiplies the current dimming-level) of color and White/TW channels as follows: colorLevel = The normalized color level in range [0..1]. White/TW is at 100%; color grows 0..100% whiteLevel = The normalized white level in range [0..1]. White/TW declines [100..0%], colors at 100% Both white and color outputs are 100% (of dimming level).
OnOff **Button** **PushButton**	INT	OnOff/Button/PushButton value given as: value = Integer value indicating the state of corresponding control input (0 or 1).

Sensors

The “sensors” property contains latest sensor information of the device, if any. This information is stored in a collection of sensor data, indexed by sensor definition name.

Following table shows content of each sensor data item.

Property	Type	Related Value
name	STR	Sensor definition name.
value	NUM	Current value.
timestamp	TS	Time when value change occurred.

Details

The “details” property remains more or less the same as shown in the datapoints REST API part but the method of receiving this data “package” differs. Thus, the same “details” data can be collected in two different ways. Firstly via “unitChanged” events as they occur (only once per state change event) and secondly via datapoints REST API requests (data is stored for 60 days).

Appendix

Value Types

Basic definition of the used storage type and formats of the values:

Value Type	Short ID	Format Description
`number`	`NUM`	Integer or floating-point quantity (generic)
`integer`	`INT`	Integer quantity (can be still provided as a floating-point value)
`percent`	`P`	(Normalized) quantity in the range [0…1] ~ 0..100%
`string`	`STR`	String value (currently limited to 24 bytes)

`boolean`	`BOOL`	Conventional representation of a Boolean value, using typically numeric 0 / non-0 contents, but generally supporting string-based representation where needed:
		`true:` `1 | "yes" | "true" | {`*anything else*`}`
		`false:` `0 | "" | "no" | "false"`
		Empty string may also be used as "undefined" / "null" value.

`timestamp`	`TS`	Floating-point value of number of seconds (double resolution, with fractional seconds) with Jan 1st, 1970 (aka Unix time). This format is used for most of timestamps other than the logging time.
`datapoint_time`	`DT`	String-based format of "`yyyymmddHHMMSS`" logging time, the representation of timestamps used for JSON export of datapoint records in the Datapoint API. In the future this format can be extended with a "`.zzz`" part – for milliseconds of the timestamp.

Property Names

This specification defines the set of standard names for logged properties, associated naming conventions, type of values and output formats. String-based ID / name of a logged property unifies access to data coming from different sources: vendor sensors (Extension Interface), DALI, Casambi-internal processes that can provide features as energy counters, diagnostics and identification properties.

Size of full name of a property and its value are limited currently to 24 bytes.

Property names conventionally can be formed hierarchically and with index-modifier parts to represent serialization of data structures into flattened list of KEY = VALUE records in the context of the unit:

Hash/object structure

```
name: {
   detailA: valueA
   detailB: valueB
}
```

yielding

```
"name.detailA" = valueA
"name.detailB" = valueB
```

Array property

```
name: [ valueA, valueB, ...]
```

yielding

```
"name:1" = valueA
"name:2" = valueB
```

Additionally, special convention is used for enumeration of properties of sub-devices (for example, DALI drivers behind a Casambi unit): indexed **"${n}"** property space for listing all properties of N-th instance of device, for example:

```
"$0.serial",   = "938577234110003"
"$0.model"      = "OSRAM OTi 85W"
"$0.address"   = "3"
...
"$1.serial",  = "583522300399545"
"$1.model"   = "XUV 35W 0.7-1.4A"
"$1.address"  = "2"
```

DALI Device Model

This part of the data model is heavily affected by the DALI specification and the recent standardized extensions for DT50/51/52 (**IEC62386-2xx**, parts 251/252/253).

The properties are provided in the context of an indexed sub-device container – a single object that represents a (DALI) device: "**`${n}.serial`**", "**`${n}.address`**", "**`${n}.model`**", etc.

The only mandatory identification key property of the device data structure is "**serial**", and the logical **"${n}"** reference does not have strict one-to-one relation with the serial ID (since it must be a short identifier), thus it is not guaranteed to be always used for the same data object. Conventionally, DALI address of the device is used as the reference.

All string properties are handled with the "`[adaptive]`" logging policy with period of 24 hours, i.e. every change matter and is logged immediately, and if the value does not change, it is logged as is again after the configured time interval. Note that these values are not expected to change usually.