


# CloudWorks Integration

## REST API User Manual

Cape Digital Solutions

August 2026

# Table of Contents


1. Introduction ..... 3
2. Getting Started ..... 3
   2.1 Base URL and Ports ..... 3
   2.2 Authentication ..... 3
   2.3 Conventions ..... 3
3. Response Codes and Error Handling ..... 4
4. Endpoint Reference ..... 4
   4.1 General Functions ..... 4
   4.2 Script Command Functions ..... 7
   4.3 Tank Table Functions ..... 8
   4.4 Zone Functions ..... 9
   4.5 Site Functions ..... 10
   4.6 User Functions ..... 12
   4.7 Device Functions ..... 14
5. getDeviceData — Application Data Reference ..... 17
   5.1 Common Response Envelope ..... 17
   5.2 Hourly Summarisation (summarise=true) ..... 17
   5.3 Application Field Reference ..... 18
6. getDeviceInformation Field Reference ..... 27
   6.1 Basic Device Fields (basicData=true) ..... 27
   6.2 Full Device Fields (basicData=false) ..... 27
7. Appendix — Value Reference ..... 29
   7.1 Status Values ..... 29
   7.2 Practical Notes ..... 29

# 1. Introduction

The CloudWorks Integration application embeds a REST API server that gives external systems full access to the CloudWorks device estate: zones, sites, devices, users, live communications status, remote commands and the complete datalog history for every supported device application.

The API is hosted inside the CloudWorks Integration application itself (ASP.NET Core / Kestrel running on a background thread). It starts automatically with the application and listens on the configured HTTP and/or HTTPS ports.

All requests and responses use JSON unless stated otherwise. The API is stateless: every request must carry the API key header described in Section 2.

# 2. Getting Started

## 2.1 Base URL and Ports

The server listens on all network interfaces (0.0.0.0). Default ports are HTTP 18500 and HTTPS 18501; both are configurable in the CloudWorks Integration System Setup screen. A port set to 0 disables that listener.

```
http://<server>:18500/api/<endpoint>
https://<server>:18501/api/<endpoint>
```

Use **getServerConfiguration** to discover whether secure (HTTPS) communications are available and which secure port to use.

## 2.2 Authentication

Every call must include the API key in the **X-API-Key** HTTP header. The key identifies the customer database the request is executed against — all data returned or modified is automatically scoped to that database.

```
X-API-Key: <your API key>
```

Requests without a valid key receive **401 Unauthorized** with no body.

## 2.3 Conventions

- GET endpoints read data; POST endpoints create, modify or delete data.
- Parameters are passed either in the URL query string or as a JSON request body — each endpoint entry in Section 4 states which.
- Save endpoints (saveZone, saveSite, saveUser) are upserts: they update the record when the key exists and insert it otherwise. All body fields must be present.
- Dates are supplied in one of the accepted formats: ISO yyyy-MM-dd[ HH:mm[:ss]] (a T separator is also accepted) or dd/MM/yyyy[ HH:mm[:ss]]. Any other date format is rejected with 400 Bad Request. Decimal values always use a dot as the decimal separator, regardless of server regional settings.
- Datalog query dates (getDeviceData, deleteDatalog) are interpreted in the site's local time zone and converted to UTC internally; returned datalog timestamps are in site local time.
- getCommsRegister returns lastSeenUTC in UTC; getDeviceData returns lastSeen converted to site local time.
- Boolean query parameters accept true / false; any other value (or an omitted parameter) is treated as false.
- Code parameters (deviceCode, siteCode, zoneCode, userCode and similar) may only contain letters, digits, dash, underscore, dot and space, up to 32 characters. Anything else is rejected with 400 Bad Request before the database is queried.

- Datalog date ranges (getDeviceData, deleteDatalog) must have startDate on or before endDate, dates between 2016 and the near future, and may span at most 400 days per request.
- All fractional numeric values in responses are rounded to 3 decimal places, except latitude / longitude values which are returned to 6 decimal places.

# 3. Response Codes and Error Handling

| Status | Meaning | Body |
|---|---|---|
| 200 OK | Request succeeded | JSON object, plain text (sendRemoteCommand) or empty (action endpoints) |
| 400 Bad Request | Invalid or missing parameter, unknown record, or a documented business error | Plain-text reason, e.g. "Device Code is blank", "Device 123 not found in the database", "deviceCode contains invalid characters", "Date range too large - maximum 400 days per request" |
| 401 Unauthorized | Missing or invalid X-API-Key header | Empty |
| 500 Problem | Unexpected server error | Problem JSON with the message "Requested command threw an unexpected exception" |

Client code should treat any non-200 response as a failure and read the body text for the reason where present.

# 4. Endpoint Reference

The API exposes 32 endpoints in seven functional groups. Each entry below lists the HTTP method and route, the parameters, the response, and an example call.

## 4.1 General Functions

### pingServer

**GET** `/api/pingServer`

Verifies that the server is reachable and that the supplied API key is valid. Use this as a connectivity / health check before making other calls.

Parameters: none (API key header only).

**Response:** 200 OK with an empty body when the server is up and the API key is valid; 401 Unauthorized when the key is missing or invalid.

```
curl -H "X-API-Key: <your key>" http://myserver:18500/api/pingServer
```

### getServerConfiguration

**GET** `/api/getServerConfiguration`

Returns the server and database connection details associated with the supplied API key. Client applications use this to discover the secure (HTTPS) port, the direct TCP communications port and the database connection settings.

Parameters: none (API key header only).

**Response:** 200 OK with a JSON Server Configuration object:

| Field | Type | Description |
|---|---|---|
| message | String | General message about the data |
| secureCommsAvailable | Boolean | True when an HTTPS port is configured on the server |
| securePort | Integer | Secure (HTTPS) API port |
| serverVersion | String | Server software version |
| serverCommsPort | Integer | Server direct TCP communications port |
| databaseServer | String | Database server name / public IP |
| databasePort | Integer | Database TCP port |
| databaseName | String | Database name for this API key |
| databaseUsername | String | Database username |
| databasePassword | String | Database password |
| databaseCode | String | Database code associated with the API key |

**Note:** The response contains live database credentials. Treat the output as sensitive and never log or display it.

```
curl -H "X-API-Key: <your key>" http://myserver:18500/api/getServerConfiguration
```

### reloadConfiguration

**POST** `/api/reloadConfiguration`

Flags a device so that it reloads its configuration the next time it contacts the server. Passing reload=false cancels a pending reload.

| Parameter | In | Required | Description |
|---|---|---|---|
| deviceCode | Query | Yes | Device code of the target device |
| reload | Query | Yes | true to request a configuration reload, false to cancel a pending reload |

**Response:** 200 OK on success; 400 Bad Request when the device code is blank.

```
curl -X POST -H "X-API-Key: <your key>"
"http://myserver:18500/api/reloadConfiguration?deviceCode=2107070002&reload=true"
```

### setAlarmAlert

**POST** `/api/setAlarmAlert`

Sets or clears the alarm and alert (warning) state of a device, together with the associated messages. Setting alarm or alert to false clears the corresponding message.

| Parameter | In | Required | Description |
|---|---|---|---|
| deviceCode | Query | Yes | Device code of the target device |
| alarm | Query | Yes | true to raise the alarm state, false to clear it |
| alarmMessage | Query | No | Message stored with the alarm state (used when alarm=true) |
| alert | Query | Yes | true to raise the alert state, false to clear it |
| alertMessage | Query | No | Message stored with the alert state (used when alert=true) |

**Response:** 200 OK on success; 400 Bad Request when the device code is blank.

```
curl -X POST -H "X-API-Key: <your key>"
"http://myserver:18500/api/setAlarmAlert?deviceCode=2107070002&alarm=true&alarmMessage=High%20Level&alert=false"
```

### sendRemoteCommand

**POST** `/api/sendRemoteCommand`

Sends a raw command directly to an online device and waits for its reply. The call blocks until the device responds or the communication timeout expires.

| Parameter | In | Required | Description |
|---|---|---|---|
| `deviceCode` | Query | Yes | Device code of the target device |
| `command` | Query | Yes | The command string to send to the device |

**Response:** 200 OK with the device's raw text response. 400 Bad Request with one of: "No Response" (timeout), "Device is currently offline", or "Device not found in Comms Register".

**Note:** The device must currently be online (visible in the communications register). Battery devices that are asleep cannot receive remote commands — queue a script command instead.

```
curl -X POST -H "X-API-Key: <your key>"
"http://myserver:18500/api/sendRemoteCommand?deviceCode=2107070002&command=STATUS"
```

### sendEmail

**POST** `/api/sendEmail`

Sends an email through the server's configured email account.

**Parameters:**

| Field | Type | Required | Description |
|---|---|---|---|
| `recipientAddress` | String | Yes | Recipient email address |
| `subject` | String | Yes | Message subject |
| `body` | String | Yes | Message body (plain text or HTML) |
| `isHTML` | Boolean | Yes | true when the body contains HTML |

**Response:** 200 OK when the email is sent; 400 Bad Request for missing fields or malformed JSON; 500 Problem when sending fails.

```
curl -X POST -H "X-API-Key: <your key>" -H "Content-Type: application/json" -d
"{\"recipientAddress\":\"user@example.com\",\"subject\":\"Test\",\"body\":\"Hello\",\"isHTML\":false}" http://myserver:18500/api/sendEmail
```

### getCommsRegister

**GET** `/api/getCommsRegister`

Returns the live communications register — the server's in-memory view of every device it is currently managing, including online status, power information, signal strength, firmware version and a per-period communications analyser (packets, drops, latency, signal).

| Parameter | In | Required | Description |
|---|---|---|---|
| deviceCode | Query | No | Device code to filter by; omit or leave blank for all devices |

**Response:** 200 OK with { message, devices[] }; 400 Bad Request when the requested device is not found. Device fields:

| Field | Type | Description |
|---|---|---|
| deviceCode | String | Unique device code |
| lockCode | String | Site code the device is locked to |
| status | String | Online / Offline / Sleep |
| applicationType | String | Application description for the device |
| deviceType | String | Device type description |
| databaseCode | String | Database code the device belongs to |
| databaseMove | String | Pending database move, if any |
| powerStatus | String | Mains / Battery — current power source |
| powerMode | String | Mains / Battery / Dual — configured power mode |
| signalStrength | Integer | Current GSM signal strength in % |
| wakeupType | String | Manual / Alarm / Auto (battery devices only) |
| firmwareVersion | String | Device firmware version |
| lastSeenUTC | Date | Last time the device was seen (UTC) |
| ipAddress | String | Current IP address of the device |
| ipPort | Integer | Current IP port of the device |
| latency | Integer | Current round-trip latency in ms |
| commsAnalyser[] | Array | Per-period analyser entries: totalPackets, droppedPackets, averageLatency, averageSignal |

```
curl -H "X-API-Key: <your key>"
"http://myserver:18500/api/getCommsRegister?deviceCode=2107070002"
```

## 4.2 Script Command Functions

### getScriptCommandInformation

**GET** /api/getScriptCommandInformation

Returns the queued offline script commands, per device, together with any responses already received and a completion flag. Script commands are delivered to battery devices the next time they wake up.

| Parameter | In | Required | Description |
|---|---|---|---|
| deviceCode | Query | No | Device code to filter by; omit for all devices |

**Response:** 200 OK with { message, devices[] } where each device holds { zoneCode, siteCode, deviceCode, commands[] } and each command holds { command, response, complete }; 400 Bad Request when no script commands are found.

```
curl -H "X-API-Key: <your key>"
"http://myserver:18500/api/getScriptCommandInformation?deviceCode=2107070002"
```

### saveScriptCommand

**POST** `/api/saveScriptCommand`

Queues a new offline script command for a device and flags the device for a configuration update so the command is collected at its next contact.

| Parameter | In | Required | Description |
|---|---|---|---|
| deviceCode | Query | Yes | Device code of the target device |
| command | Query | Yes | The command string to queue |

**Response:** 200 OK on success; 400 Bad Request for a missing device code or command.

```
curl -X POST -H "X-API-Key: <your key>"
"http://myserver:18500/api/saveScriptCommand?deviceCode=2107070002&command=SET%20WAKEUP%2060"
```

### clearScriptCommands

**POST** `/api/clearScriptCommands`

Deletes all queued script commands for a device.

| Parameter | In | Required | Description |
|---|---|---|---|
| deviceCode | Query | Yes | Device code of the target device |

**Response:** 200 OK on success; 400 Bad Request when the device code is missing.

```
curl -X POST -H "X-API-Key: <your key>"
"http://myserver:18500/api/clearScriptCommands?deviceCode=2107070002"
```

## 4.3 Tank Table Functions

### getTankTable

**GET** `/api/getTankTable`

Returns the level-to-volume calibration table for a tank device, ordered by product level.

| Parameter | In | Required | Description |
|---|---|---|---|
| deviceCode | Query | Yes | Device code of the tank device |

**Response:** 200 OK with { message, deviceCode, tankTable[] } where each entry holds { level, volume, capturedEntry }. capturedEntry marks points captured from a live calibration rather than typed in.

```
curl -H "X-API-Key: <your key>" "http://myserver:18500/api/getTankTable?deviceCode=2107070002"
```

### saveTankTable

**POST** `/api/saveTankTable`

Replaces the entire tank table for a device. The existing table is deleted and the supplied entries are inserted. Levels are stored to 3 decimal places and volumes to 4.

| Parameter | In | Required | Description |
|---|---|---|---|
| deviceCode | Query | Yes | Device code of the tank device |

**Request body:**

```
[ { "level": 0.5, "volume": 1.25, "capturedEntry": false }, ... ]
```

**Response:** 200 OK on success; 400 Bad Request for a missing device code or malformed body.

**Note:** This is a full replace — always send the complete table, not just changed rows.

```
curl -X POST -H "X-API-Key: <your key>" -H "Content-Type: application/json" -d
"[{\"level\":0.5,\"volume\":1.25,\"capturedEntry\":false}]"
"http://myserver:18500/api/saveTankTable?deviceCode=2107070002"
```

### calibrateTank

POST /api/calibrateTank

Calibrates the level reading of a tank device fitted with a hydrostatic (pressure) sensor. The dipped level is queued as a script command and delivered to the device at its next contact; the device uses it to recalculate its pressure-to-level conversion factor, calibrating out the product density.

**Parameters:** deviceCode (query, required) - device to calibrate; dipLevel (query, required) - dipped product level in metres, measured from the bottom of the tank; must be greater than 0.

**Response:** 200 OK when the calibration command has been queued; 400 Bad Request for a blank or invalid device code, or a dip level of zero or less.

**Note:** Like all script commands the calibration takes effect when the device next contacts the server - for battery devices that is the next wakeup. Dip the tank as close to the queueing time as possible so the level has not changed when the command is applied.

```
curl -X POST -H "X-API-Key: <your key>"
"http://myserver:18500/api/calibrateTank?deviceCode=2107070002&dipLevel=1.200"
```

## 4.4 Zone Functions

### getZoneInformation

**GET** /api/getZoneInformation

Returns zone records, including live alarm and alert roll-ups (a zone is flagged when any device inside it is in alarm / alert) and the zone contact details.

| Parameter | In | Required | Description |
|---|---|---|---|
| zoneCode | Query | No | Zone code to filter by; omit for all zones |

**Response:** 200 OK with { message, zones[] }; 400 Bad Request when the zone is not found. Zone fields:

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| zoneCode | String | Zone code |
| alarm | Boolean | True when any device in the zone is in alarm |
| alert | Boolean | True when any device in the zone is in alert |
| description | String | Zone description |
| contactName | String | Contact name |
| address1..address4 | String | Address lines 1 to 4 |
| country | String | Country |
| postalCode | String | Postal code |
| telephone | String | Telephone number |
| cellular | String | Cellular number |
| emailAddress | String | Email address |

```
curl -H "X-API-Key: <your key>" "http://myserver:18500/api/getZoneInformation?zoneCode=ZN01"
```

### saveZone

**POST** /api/saveZone

Creates a new zone or updates an existing one (matched on zoneCode). All body fields must be present.

**JSON request body fields:** zoneCode (upsert key), description, contactName, address1–address4, country, postalCode, telephone, cellular, emailAddress — all Strings, all required.

**Response:** 200 OK on success; 400 Bad Request for a blank zone code or malformed body.

```
curl -X POST -H "X-API-Key: <your key>" -H "Content-Type: application/json" -d @zone.json
http://myserver:18500/api/saveZone
```

### deleteZone

**POST** /api/deleteZone

Deletes a zone.

| Parameter | In | Required | Description |
|---|---|---|---|
| zoneCode | Query | Yes | Zone code to delete |

**Response:** 200 OK on success; 400 Bad Request for a blank zone code.

**Warning:** This cascades: all sites in the zone and all devices at those sites are deleted as well. The operation cannot be undone.

```
curl -X POST -H "X-API-Key: <your key>" "http://myserver:18500/api/deleteZone?zoneCode=ZN01"
```

## 4.5 Site Functions

### getSiteInformation

**GET** /api/getSiteInformation

Returns site records, including live alarm / alert roll-ups, contact details and the site time zone.

| Parameter | In | Required | Description |
|---|---|---|---|
| siteCode | Query | No | Site code to filter by; omit for all sites |

**Response:** 200 OK with { message, sites[] }. Site fields match the zone fields plus zoneCode (parent zone) and timeZone (site time zone identifier); 400 Bad Request when the site is not found.

```
curl -H "X-API-Key: <your key>" "http://myserver:18500/api/getSiteInformation?siteCode=ST01"
```

### saveSite

**POST** /api/saveSite

Creates a new site or updates an existing one (matched on siteCode). All body fields must be present.

**JSON request body fields:** siteCode (upsert key), description, contactName, address1–address4, country, postalCode, telephone, cellular, emailAddress, timeZone — all Strings, all required.

**Response:** 200 OK on success; 400 Bad Request for a blank site code or malformed body.

```
curl -X POST -H "X-API-Key: <your key>" -H "Content-Type: application/json" -d @site.json
http://myserver:18500/api/saveSite
```

### deleteSite

**POST** /api/deleteSite

Deletes a site.

| Parameter | In | Required | Description |
|---|---|---|---|
| siteCode | Query | Yes | Site code to delete |

**Response:** 200 OK on success; 400 Bad Request for a blank site code.

**Warning:** This cascades: all devices at the site are deleted as well. The operation cannot be undone.

```
curl -X POST -H "X-API-Key: <your key>" "http://myserver:18500/api/deleteSite?siteCode=ST01"
```

### moveSite

**POST** /api/moveSite

Moves a site (and therefore all of its devices) to a different zone.

| Parameter | In | Required | Description |
|---|---|---|---|
| siteCode | Query | Yes | Site code to move |
| destinationZoneCode | Query | Yes | Zone code the site should move to |

**Response:** 200 OK on success; 400 Bad Request for blank parameters.

```
curl -X POST -H "X-API-Key: <your key>"
"http://myserver:18500/api/moveSite?siteCode=ST01&destinationZoneCode=ZN02"
```

## 4.6 User Functions

### getUserInformation

**GET** `/api/getUserInformation`

Returns system user records.

| Parameter | In | Required | Description |
|---|---|---|---|
| userCode | Query | No | User code to filter by; omit for all users |

**Response:** 200 OK with { message, users[] }; 400 Bad Request when the user is not found. User fields:

| Field | Type | Description |
|---|---|---|
| userCode | String | User code (login name) |
| password | String | Encrypted password as stored |
| userType | String | User type code: N=Normal, S=Supervisor, O=Operator, T=Technician, A=Administrator, Z=Zone User, U=Site User, D=Device User |
| accessCode | String | Access code (zone / site / device the user is restricted to, per user type) |
| name | String | Full name |
| address1..address4 | String | Address lines |
| country | String | Country |
| postalCode | String | Postal code |
| telephone | String | Telephone number |
| cellular | String | Cellular number |
| emailAddress | String | Email address |

```
curl -H "X-API-Key: <your key>" "http://myserver:18500/api/getUserInformation?userCode=jsmith"
```

### saveUser

**POST** `/api/saveUser`

Creates a new user or updates an existing one (matched on userCode). When creating a user a random 10 character password is generated; use resetUserPassword with newUser=true afterwards to email the credentials to the user.

**JSON request body fields:**

| Field | Type | Required | Description |
|---|---|---|---|
| userCode | String | Yes | User code (upsert key) |
| userType | String | Yes | One of: Normal, Supervisor, Operator, Technician, Administrator, Zone User, Site User, Device User |
| accessCode | String | Yes | Access restriction code for zone / site / device scoped user types |
| name | String | Yes | Full name |
| address1 - address4 | String | Yes | Address lines |
| country | String | Yes | Country |

| | | | |
|---|---|---|---|
| postalCode | String | Yes | Postal code |
| telephone | String | Yes | Telephone number |
| cellular | String | Yes | Cellular number |
| emailAddress | String | Yes | Email address (needed for password emails) |

**Response:** 200 OK on success; 400 Bad Request for a blank user code or malformed body.

```
curl -X POST -H "X-API-Key: <your key>" -H "Content-Type: application/json" -d @user.json
http://myserver:18500/api/saveUser
```

### deleteUser

**POST** /api/deleteUser

Deletes a system user.

| Parameter | In | Required | Description |
|---|---|---|---|
| userCode | Query | Yes | User code to delete |

**Response:** 200 OK on success; 400 Bad Request for a blank user code.

```
curl -X POST -H "X-API-Key: <your key>" "http://myserver:18500/api/deleteUser?userCode=jsmith"
```

### resetUserPassword

**POST** /api/resetUserPassword

Generates a new random 10 character password for the user, stores it (encrypted) and emails the user their new credentials together with a pre-configured .cwf connection shortcut file. With newUser=true a welcome email is sent instead of a password-reset email.

| Parameter | In | Required | Description |
|---|---|---|---|
| userCode | Query | Yes | User code to reset |
| newUser | Query | No | true to send the new-user welcome email instead of the reset email |

**Response:** 200 OK with { message: "<new password>" } when the email is sent; 400 with the password in the problem detail when the email fails; 400 Bad Request when the user does not exist.

**Note:** The user record must contain a valid email address.

```
curl -X POST -H "X-API-Key: <your key>"
"http://myserver:18500/api/resetUserPassword?userCode=jsmith&newUser=false"
```

### changeUserPassword

**POST** /api/changeUserPassword

Sets a user's password to a value already encrypted by the client using the system password encryption.

| Parameter | In | Required | Description |
|---|---|---|---|

| | | | |
|---|---|---|---|
| userCode | Query | Yes | User code |
| newEncryptedPassword | Query | Yes | The new password, pre-encrypted with the system password encryption |

**Response:** 200 OK on success; 400 Bad Request when the user does not exist.

```
curl -X POST -H "X-API-Key: <your key>"
"http://myserver:18500/api/changeUserPassword?userCode=jsmith&newEncryptedPassword=<encrypted>"
```

## 4.7 Device Functions

### getDeviceInformation

**GET** /api/getDeviceInformation

Returns device records. With basicData=true only the core identity / status fields are returned; otherwise every telemetry column in the device table is included (totalisers, loops, pressures, flow, meter data, soil data, etc.).

| Parameter | In | Required | Description |
|---|---|---|---|
| deviceCode | Query | No | Device code to filter by; omit for all devices |
| basicData | Query | No | true to return the reduced basic field set |

**Response:** 200 OK with { message, devices[] }; 400 Bad Request when the device is not found. See Section 6 for the full and basic device field lists.

```
curl -H "X-API-Key: <your key>"
"http://myserver:18500/api/getDeviceInformation?deviceCode=2107070002&basicData=true"
```

### addDevice

**POST** /api/addDevice

Creates a new device record at a site. The device is created with default settings; its details are populated when it first connects to the server.

**JSON request body fields:** siteCode (query, required) - site the device is added to; deviceCode (query, required) - unique code of the new device.

**Response:** 200 OK on success; 400 Bad Request for a blank site or device code, or when the device already exists ("Device already in Site - <site code>").

```
curl -X POST -H "X-API-Key: <your key>"
"http://myserver:18500/api/addDevice?siteCode=ST01&deviceCode=2304130034"
```

### deleteDevice

**POST** /api/deleteDevice

Deletes a device record.

| Parameter | In | Required | Description |
|---|---|---|---|
| deviceCode | Query | Yes | Device code to delete |

**Response:** 200 OK on success; 400 Bad Request for a blank device code.

**Warning:** The operation cannot be undone.

```
curl -X POST -H "X-API-Key: <your key>"
"http://myserver:18500/api/deleteDevice?deviceCode=2107070002"
```

### moveDevice

**POST** `/api/moveDevice`

Moves a device to a different site.

| Parameter | In | Required | Description |
|---|---|---|---|
| `deviceCode` | Query | Yes | Device code to move |
| `destinationSiteCode` | Query | Yes | Site code the device should move to |

**Response:** 200 OK on success; 400 Bad Request for blank parameters.

```
curl -X POST -H "X-API-Key: <your key>"
"http://myserver:18500/api/moveDevice?deviceCode=2107070002&destinationSiteCode=ST02"
```

### replaceDevice

**POST** `/api/replaceDevice`

Replaces the hardware for an installation: the existing device record is re-keyed from the old device code to the new device code, keeping the site assignment, configuration and datalog history.

| Parameter | In | Required | Description |
|---|---|---|---|
| `oldDeviceCode` | Query | Yes | Device code being replaced |
| `newDeviceCode` | Query | Yes | Device code of the replacement hardware |

**Response:** 200 OK on success; 400 Bad Request for blank parameters.

```
curl -X POST -H "X-API-Key: <your key>"
"http://myserver:18500/api/replaceDevice?oldDeviceCode=2107070002&newDeviceCode=2304130034"
```

### deleteDatalog

**POST** `/api/deleteDatalog`

Deletes datalog entries for a device between two dates. Dates are supplied in the site's local time and converted to UTC internally.

| Parameter | In | Required | Description |
|---|---|---|---|
| `deviceCode` | Query | Yes | Device code |
| `startDate` | Query | Yes | Start of the range (site local time, e.g. 2026-07-01 00:00:00) |
| `endDate` | Query | Yes | End of the range (site local time) |

**Response:** 200 OK on success; 400 Bad Request for an unknown device, an invalid device code, or an invalid date or date range.

**Warning:** Deleted datalog entries cannot be recovered.

```
curl -X POST -H "X-API-Key: <your key>"
"http://myserver:18500/api/deleteDatalog?deviceCode=2107070002&startDate=2026-07-01&endDate=2026-07-10"
```

### getDeviceData

**GET** `/api/getDeviceData`

The main data retrieval call. Returns the device's current values plus its datalog readings between two dates. The shape of the response depends on the device's application code — see Section 5 for the response envelope and every application-specific field set. With summarise=true the datalog is reduced to hourly readings.

| Parameter | In | Required | Description |
|---|---|---|---|
| `deviceCode` | Query | Yes | Device code of the datalogger |
| `startDate` | Query | Yes | Datalog start date (site local time; accepted formats and range limits per Section 2.3) |
| `endDate` | Query | Yes | Datalog end date (site local time) |
| `summarise` | Query | No | true to summarise the data into hourly readings |

**Response:** 200 OK with the application-specific JSON object (Section 5); 400 Bad Request for an unknown device, an invalid device code, an invalid date or date range, or "Unsupported application type".

```
curl -H "X-API-Key: <your key>"
"http://myserver:18500/api/getDeviceData?deviceCode=2107070002&startDate=2026-07-01&endDate=2026-07-10&summarise=true"
```

# 5. getDeviceData — Application Data Reference

Every device runs one of the CloudWorks device applications, identified by its application code (returned as applicationCode by getDeviceInformation). The getDeviceData response is a single JSON object whose fields depend on that application. Every response, regardless of application, starts with the same common envelope of device fields, followed by the application-specific current values and a datalog[] array of logged readings.

Devices with an application code not listed below return 400 "Unsupported application type".

## 5.1 Common Response Envelope

These fields appear in every getDeviceData response:

| Field | Type | Description |
|---|---|---|
| message | String | Describes the data and the date range returned |
| commsStatus | String | Online / Offline / Sleep / Unknown |
| lastSeen | Date | Last time the device was seen (site local time) |
| timeZone | String | Site time zone display name |
| siteDescription | String | "<Site Code> : <Site Description>" |
| deviceCode | String | Unique device code |
| deviceType | String | Device type description, e.g. CDS552 - Cirrus Loop |
| applicationCode | String | Application code, e.g. 21 |
| dataIndex | Integer | Index key into the datalog table |
| description | String | Device description |
| serialNumber | String | Serial number |
| gsmSignal | Integer | GSM signal strength in % |
| batteryStatus | Integer | Battery life remaining in %; reported for all readings regardless of power source |
| powerStatus | String | Mains / Battery / Unknown — current power source |
| powerMode | String | Mains / Battery / Dual — configured power mode |
| alarm | Boolean | Alarm active flag |
| alert | Boolean | Alert / warning active flag |
| alarmMessage | String | Alarm message |
| alertMessage | String | Alert / warning message |
| latitude | Double | Device latitude (6 decimal places) |
| longitude | Double | Device longitude (6 decimal places) |
| wakeupPeriod | Integer | Wakeup period in minutes |
| datalogPeriod | Integer | Datalog period in minutes |
| lastDatalogDate | Date | Date / time of the last datalog entry (site local time) |
| lastFtpUpload | Date | Last FTP upload time (site local time) |
| userConfigurationData | String | User defined configuration data (max 220 bytes) |

## 5.2 Hourly Summarisation (summarise=true)

When summarise=true is supplied, the raw datalog is reduced to one reading per hour:

- The period runs from the first whole hour at or after the first reading to the last whole hour at or before the final reading.
- Each hourly entry takes the last reading that falls inside that hour.
- Hours with no readings are filled by linear interpolation between the surrounding real readings and flagged with interpolatedEntry=true.
- Trailing hours with no later real reading carry the last known values forward, also flagged as interpolated.

## 5.3 Application Field Reference

For each application below, **Current values** lists the live fields appended to the common envelope, and **Datalog entry** lists the fields of each element in the datalog[] array. Every datalog entry also includes logDate (datalog date/time, local time at site). Application code 036 is not implemented.

### Application 000

Dual pulse counting with a 4-20mA loop input and digital input/output.

**Current values:** totaliser1 / totaliser1Units, totaliser2 / totaliser2Units, loopReading / loopUnits, input, output

**Datalog entry:** logDate, totaliser1, totaliser2, consumption1, consumption2, loopReading, input, output, batteryStatus, powerStatus, interpolatedEntry

### Application 001

Dual pulse metering with pressure on the 4-20mA loop; flow is calculated from consumption.

**Current values:** totaliser1 / totaliser1Units, totaliser2 / totaliser2Units, loopReading (pressure) / loopUnits, input, output

**Datalog entry:** logDate, totaliser1, totaliser2, consumption (combined), flow (calculated), pressure, input, output, batteryStatus, powerStatus, interpolatedEntry

### Application 002

Forward / reverse pulse metering with pressure on the 4-20mA loop.

**Current values:** forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits, loopReading (pressure) / loopUnits, input, output

**Datalog entry:** logDate, forwardTotaliser, reverseTotaliser, forwardConsumption, reverseConsumption, flow (calculated), pressure, input, output, batteryStatus, powerStatus, interpolatedEntry

### Application 003

Serial/interface meter with forward and reverse totalisers, live flow and pressure readings.

**Current values:** interfaceError, forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits, flow / flowUnits, pressure / pressureUnits, loopReading / loopUnits, input, output

**Datalog entry:** logDate, interfaceError, forwardTotaliser, reverseTotaliser, forwardConsumption, reverseConsumption, flow, pressure, loopReading, input, output, batteryStatus, powerStatus, interpolatedEntry

### Application 004

ECO-interface meter plus dual pulse totalisers, loop and digital I/O.

**Current values:** interfaceError, forwardTotaliser / forwardTotaliserUnits, medium (MBus code), meterSerial, totaliser1 / totaliser1Units, totaliser2 / totaliser2Units, loopReading / loopUnits, input, output

**Datalog entry:** logDate, ecoStatus, serialNumber, interfaceError, totaliser (ECO), consumption, flow (calculated), totaliser1, totaliser2, loopReading, input, output, batteryStatus, powerStatus, interpolatedEntry

### Application 005

Serial/interface meter (same data set as application 003).

**Current values:** interfaceError, forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits, flow / flowUnits, pressure / pressureUnits, loopReading / loopUnits, input, output

**Datalog entry:** logDate, interfaceError, forwardTotaliser, reverseTotaliser, forwardConsumption, reverseConsumption, flow, pressure, loopReading, input, output, batteryStatus, powerStatus, interpolatedEntry

### Application 006

MBus meter with cumulative / forward / reverse totalisers, temperatures and meter diagnostics, plus loop and digital I/O.

**Current values:** interfaceError, meterBattery (days), cumulativeTotaliser / cumulativeTotaliserUnits, forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits, medium (MBus code), meterSerial, mbusStatus, flow / flowUnits, ambientTemperature / ambientTemperatureUnits, mediumTemperature / mediumTemperatureUnits, meterAlarmData, meterData, meterFirmwareVersion, loopReading / loopUnits, input, output

**Datalog entry:** logDate, mbusStatus, serialNumber, interfaceError, cumulativeTotaliser, cumulativeConsumption, forwardTotaliser, forwardConsumption, reverseTotaliser, reverseConsumption, flow, mediumTemperature, ambientTemperature, meterBatteryLife (days), meterAlarmData, meterData, loopReading, input, output, batteryStatus, powerStatus, interpolatedEntry

### Application 007

Simple dual pulse totaliser logging.

**Current values:** totaliser1 / totaliser1Units, totaliser2 / totaliser2Units

**Datalog entry:** logDate, totaliser1, totaliser2, consumption1, consumption2, batteryStatus, powerStatus, interpolatedEntry

### Application 008

Dual pulse totalisers with combined consumption and calculated flow.

**Current values:** totaliser1 / totaliser1Units, totaliser2 / totaliser2Units

**Datalog entry:** logDate, totaliser1, totaliser2, consumption (combined), flow (calculated), batteryStatus, powerStatus, interpolatedEntry

### Application 009

Forward / reverse pulse totalisers with calculated flow.

**Current values:** forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits

**Datalog entry:** logDate, forwardTotaliser, reverseTotaliser, forwardConsumption, reverseConsumption, flow (calculated), batteryStatus, powerStatus, interpolatedEntry

### Application 010

ECO-interface meter plus dual pulse totalisers (no loop / digital I/O).

**Current values:** interfaceError, forwardTotaliser / forwardTotaliserUnits, medium (MBus code), meterSerial, totaliser1 / totaliser1Units, totaliser2 / totaliser2Units

**Datalog entry:** logDate, ecoStatus, serialNumber, interfaceError, totaliser (ECO), consumption, flow (calculated), totaliser1, totaliser2, batteryStatus, powerStatus, interpolatedEntry

### Application 011

MBus meter with full totaliser set, temperatures and meter diagnostics (no loop / digital I/O).

**Current values:** interfaceError, meterBattery (days), cumulativeTotaliser / cumulativeTotaliserUnits, forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits, medium (MBus code), meterSerial, mbusStatus, flow / flowUnits, ambientTemperature / ambientTemperatureUnits, mediumTemperature / mediumTemperatureUnits, meterAlarmData, meterData, meterFirmwareVersion

**Datalog entry:** logDate, mbusStatus, serialNumber, interfaceError, cumulativeTotaliser, cumulativeConsumption, forwardTotaliser, forwardConsumption, reverseTotaliser, reverseConsumption, flow, mediumTemperature, ambientTemperature, meterBatteryLife (days), meterAlarmData, meterData, batteryStatus, powerStatus, interpolatedEntry

### Application 012

Single 4-20mA loop sensor with high / low alarms.

**Current values:** loopReading / loopUnits, loopAlarm (None / High Alarm / Low Alarm)

**Datalog entry:** logDate, loopReading, loopAlarm, batteryStatus, powerStatus, interpolatedEntry

### Application 013

Serial/interface meter with loop alarm monitoring.

**Current values:** interfaceError, forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits, flow / flowUnits, pressure / pressureUnits, loopReading / loopUnits, loopAlarm

**Datalog entry:** logDate, interfaceError, forwardTotaliser, reverseTotaliser, forwardConsumption, reverseConsumption, flow, pressure, loopReading, loopAlarm, batteryStatus, powerStatus, interpolatedEntry

### Application 014

Serial/interface meter with loop alarm monitoring (same data set as application 013).

**Current values:** interfaceError, forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits, flow / flowUnits, pressure / pressureUnits, loopReading / loopUnits, loopAlarm

**Datalog entry:** logDate, interfaceError, forwardTotaliser, reverseTotaliser, forwardConsumption, reverseConsumption, flow, pressure, loopReading, loopAlarm, batteryStatus, powerStatus, interpolatedEntry

### Application 015

Dual pulse totalisers with a monitored 4-20mA loop.

**Current values:** totaliser1 / totaliser1Units, totaliser2 / totaliser2Units, loopReading / loopUnits, loopAlarm

**Datalog entry:** logDate, totaliser1, totaliser2, consumption1, consumption2, loopReading, loopAlarm, batteryStatus, powerStatus, interpolatedEntry

### Application 016

Dual pulse totalisers with pressure on a monitored loop; combined consumption and calculated flow.

**Current values:** totaliser1 / totaliser1Units, totaliser2 / totaliser2Units, loopReading (pressure) / loopUnits, loopAlarm

**Datalog entry:** logDate, totaliser1, totaliser2, consumption (combined), flow (calculated), pressure, loopAlarm, batteryStatus, powerStatus, interpolatedEntry

### Application 017

Forward / reverse pulse totalisers with pressure on a monitored loop.

**Current values:** forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits, loopReading (pressure) / loopUnits, loopAlarm

**Datalog entry:** logDate, forwardTotaliser, reverseTotaliser, forwardConsumption, reverseConsumption, flow (calculated), pressure, loopAlarm, batteryStatus, powerStatus, interpolatedEntry

### Application 018

Serial/interface meter with flow, loop and digital I/O (no pressure).

**Current values:** interfaceError, forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits, flow / flowUnits, loopReading / loopUnits, input, output

**Datalog entry:** logDate, interfaceError, forwardTotaliser, reverseTotaliser, forwardConsumption, reverseConsumption, flow, loopReading, input, output, batteryStatus, powerStatus, interpolatedEntry

### Application 019

Serial/interface meter with flow and a monitored loop.

**Current values:** interfaceError, forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits, flow / flowUnits, loopReading / loopUnits, loopAlarm

**Datalog entry:** logDate, interfaceError, forwardTotaliser, reverseTotaliser, forwardConsumption, reverseConsumption, flow, loopReading, loopAlarm, batteryStatus, powerStatus, interpolatedEntry

### Application 020

Tank level / volume monitoring with digital I/O.

**Current values:** productLevel / productLevelUnits (m), productVolume / productVolumeUnits (m3), productLevelPercentage, input, output

**Datalog entry:** logDate, productLevel, productVolume, input, output, batteryStatus, powerStatus, interpolatedEntry

### Application 021

Tank level / volume monitoring from a loop sensor with high / low alarms.

**Current values:** productLevel / productLevelUnits (m), productVolume / productVolumeUnits (m3), productLevelPercentage, loopAlarm

**Datalog entry:** logDate, productLevel, productVolume, loopAlarm, batteryStatus, powerStatus, interpolatedEntry

### Application 022

Tank level / volume monitoring via a serial interface with digital I/O.

**Current values:** interfaceError, productLevel / productLevelUnits (m), productVolume / productVolumeUnits (m3), productLevelPercentage, input, output

**Datalog entry:** logDate, interfaceError, productLevel, productVolume, input, output, batteryStatus, powerStatus, interpolatedEntry

### Application 023

Tank level / volume monitoring via a serial interface with loop alarms.

**Current values:** interfaceError, productLevel / productLevelUnits (m), productVolume / productVolumeUnits (m3), productLevelPercentage, loopAlarm

**Datalog entry:** logDate, interfaceError, productLevel, productVolume, loopAlarm, batteryStatus, powerStatus, interpolatedEntry

### Application 024

GPS position tracking with a loop input and digital I/O.

**Current values:** loopReading / loopUnits, input, output

**Datalog entry:** logDate, gpsLatitude, gpsLongitude, loopReading, gpsValidFix, gpsError, input, output (no battery / power / interpolated fields)

### Application 025

Krohne meter counters with flow, loop and digital I/O.

**Current values:** interfaceError, counter1 (m3), counter2 (m3), flow / flowUnits, loopReading / loopUnits, input, output

**Datalog entry:** logDate, counter1, counter2, consumption1, consumption2, flow, loopReading, input, output, interfaceError, batteryStatus, powerStatus, interpolatedEntry

### Application 026

Krohne meter counters with flow and a monitored loop.

**Current values:** interfaceError, counter1 (m3), counter2 (m3), flow / flowUnits, loopReading / loopUnits, loopAlarm

**Datalog entry:** logDate, counter1, counter2, consumption1, consumption2, flow, loopReading, loopAlarm, interfaceError, batteryStatus, powerStatus, interpolatedEntry

### Application 027

Dual pulse totalisers with a dedicated pressure sensor and digital I/O.

**Current values:** totaliser1 / totaliser1Units, totaliser2 / totaliser2Units, pressure / pressureUnits, input, output

**Datalog entry:** logDate, totaliser1, totaliser2, consumption (combined), flow (calculated), pressure, input, output, batteryStatus, powerStatus, interpolatedEntry

### Application 028

Interface meter dual totalisers (m3) with flow and a monitored loop.

**Current values:** interfaceError, totaliser1 (m3), totaliser2 (m3), flow / flowUnits, loopReading / loopUnits, loopAlarm

**Datalog entry:** logDate, totaliser1, totaliser2, consumption1, consumption2, flow, loopReading, loopAlarm, interfaceError, batteryStatus, powerStatus, interpolatedEntry

### Application 029

Single totaliser with two independent monitored 4-20mA loops.

**Current values:** totaliser / totaliserUnits, loop1Reading / loop1Units, loop1Alarm, loop2Reading / loop2Units, loop2Alarm

**Datalog entry:** logDate, totaliser, consumption, loop1Reading, loop1Alarm, loop2Reading, loop2Alarm, batteryStatus, powerStatus, interpolatedEntry

### Application 030

MBus meter with full totaliser set, temperatures and meter diagnostics (same data set as application 011).

**Current values:** interfaceError, meterBattery (days), cumulativeTotaliser / cumulativeTotaliserUnits, forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits, medium (MBus code), meterSerial, mbusStatus, flow / flowUnits, ambientTemperature / ambientTemperatureUnits, mediumTemperature / mediumTemperatureUnits, meterAlarmData, meterData, meterFirmwareVersion

**Datalog entry:** logDate, mbusStatus, serialNumber, interfaceError, cumulativeTotaliser, cumulativeConsumption, forwardTotaliser, forwardConsumption, reverseTotaliser, reverseConsumption, flow, mediumTemperature, ambientTemperature, meterBatteryLife (days), meterAlarmData, meterData, batteryStatus, powerStatus, interpolatedEntry

### Application 031

PRV monitoring: totaliser plus upstream / downstream pressures with per-loop alarms.

**Current values:** totaliser / totaliserUnits, pressureUpstream / pressureUpstreamUnits, loop1Alarm, pressureDownstream / pressureDownstreamUnits, loop2Alarm

**Datalog entry:** logDate, totaliser, consumption, pressureUpstream, loop1Alarm, pressureDownstream, loop2Alarm, batteryStatus, powerStatus, interpolatedEntry

### Application 032

Interface water meter with totaliser, meter identity and a monitored loop.

**Current values:** interfaceError, totaliser / totaliserUnits, meterSerial, meterSize, loopReading / loopUnits, loopAlarm

**Datalog entry:** logDate, totaliser, consumption, flow (calculated), meterSerial, loopReading, loopAlarm, interfaceError, batteryStatus, powerStatus, interpolatedEntry

### Application 033

Net / forward / reverse totalisers with flow, pressure, device alarm data and a monitored loop.

**Current values:** interfaceError, totaliser (net) / totaliserUnits, forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits, flow / flowUnits, pressure / pressureUnits, deviceAlarmData, loopReading / loopUnits, loopAlarm

**Datalog entry:** logDate, interfaceError, totaliser (net), reverseTotaliser, consumption (net), reverseConsumption, flow, pressure, loopReading, loopAlarm, batteryStatus, powerStatus, interpolatedEntry

### Application 034

Aquamaster 4 flow meter with alarms, meter voltages and a monitored loop.

**Current values:** interfaceError, forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits, flow / flowUnits, pressure / pressureUnits, meterExternalVoltage, meterBatteryVoltage, meterAlarmData (Aquamaster 4 alarm bitmask), loopReading / loopUnits, loopAlarm

**Datalog entry:** logDate, interfaceError, forwardTotaliser, reverseTotaliser, forwardConsumption, reverseConsumption, flow, pressure, alarmByte (Aquamaster 4 alarm bitmask), loopReading, loopAlarm, batteryStatus, powerStatus, interpolatedEntry

### Application 035

Interface water meter with totaliser, meter identity / medium and a monitored loop.

**Current values:** interfaceError, totaliser / totaliserUnits, meterSerial, meterMedium, meterSize, loopReading / loopUnits, loopAlarm

**Datalog entry:** logDate, totaliser, consumption, flow (calculated), meterSerial, loopReading, loopAlarm, interfaceError, batteryStatus, powerStatus, interpolatedEntry

### Application 037

Soil condition monitoring: temperature, moisture and electrical conductivity.

**Current values:** interfaceError, soilTemperature / soilTemperatureUnits, soilMoisture / soilMoistureUnits, soilElectricalConductivity / soilElectricalConductivityUnits

**Datalog entry:** logDate, interfaceError, soilTemperature, soilMoisture, soilElectricalConductivity, batteryStatus, powerStatus, interpolatedEntry

### Application 038

MAG8000 flow meter with dual totalisers, meter battery and a monitored loop.

**Current values:** interfaceError, totaliser1 / totaliser1Units, totaliser2 / totaliser2Units, flow / flowUnits, meterBattery (%), meterAlarmData (MAG8000 alarm bitmask), loopReading / loopUnits, loopAlarm

**Datalog entry:** logDate, interfaceError, totaliser1, totaliser2, consumption1, consumption2, flow, alarmByte (MAG8000 alarm bitmask), loopReading, loopAlarm, batteryStatus, powerStatus, interpolatedEntry

### Application 039

Aquamaster 3 flow meter with power-source diagnostics and a monitored loop.

**Current values:** interfaceError, forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits, flow / flowUnits, pressure / pressureUnits, meterPowerSource, meterExternalPowerStatus, meterInternalPowerStatus, meterAlarmData (Aquamaster 3 alarm bitmask), loopReading / loopUnits, loopAlarm

**Datalog entry:** logDate, interfaceError, forwardTotaliser, reverseTotaliser, forwardConsumption, reverseConsumption, flow, pressure, alarmByte (Aquamaster 3 alarm bitmask), loopReading, loopAlarm, batteryStatus, powerStatus, interpolatedEntry

### Application 040

Pulse line status monitoring with a monitored loop.

**Current values:** pulse1Status (0=Low 1=High), pulse2Status (0=Low 1=High), loopReading / loopUnits, loopAlarm

**Datalog entry:** logDate, pulse1Status, pulse2Status, loopReading, loopAlarm, batteryStatus, powerStatus, interpolatedEntry

### Application 041

Pressure monitoring via a serial interface.

**Current values:** interfaceError, pressure / pressureUnits

**Datalog entry:** logDate, pressure, interfaceError, batteryStatus, powerStatus, interpolatedEntry

### Application 042

Cathodic protection monitoring.

**Current values:** cathodicVoltage, cathodicStatus (N=None, C=Corroding, O=Overprotected)

**Datalog entry:** logDate, cathodicVoltage, cathodicStatus, batteryStatus, powerStatus, interpolatedEntry

### Application 043

QT/W803 flow meter with pressure and meter battery.

**Current values:** interfaceError, forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits, flow / flowUnits, pressure / pressureUnits, meterBattery (%), meterAlarmData (QT/W803 alarm bitmask)

**Datalog entry:** logDate, interfaceError, forwardTotaliser, reverseTotaliser, forwardConsumption, reverseConsumption, flow, pressure, alarmByte (QT/W803 alarm bitmask), batteryStatus, powerStatus, interpolatedEntry

### Application 044

Tank level / volume / temperature monitoring via a serial interface.

**Current values:** interfaceError, productLevel / productLevelUnits, productVolume / productVolumeUnits, productTemperature / productTemperatureUnits, productLevelPercentage

**Datalog entry:** logDate, level, volume, temperature, batteryStatus, powerStatus, interfaceError, interpolatedEntry

### Application 045

Interface flow meter with net / forward / reverse totalisers, flow, pressure, decoded meter alarms and a monitored 4-20mA loop.

**Current values:** interfaceError, totaliser (net) / totaliserUnits, forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits, flow / flowUnits, pressure / pressureUnits, deviceAlarmData (raw meter alarm value), deviceAlarm (decoded: blank = none / Empty Pipe / Meter Battery Low / Unknown Alarm), loopReading / loopUnits, loopAlarm

**Datalog entry:** logDate, interfaceError, forwardTotaliser, reverseTotaliser, forwardConsumption, reverseConsumption, flow, pressure, loopReading, loopAlarm, deviceAlarm, batteryStatus, powerStatus, interpolatedEntry

### Application 046

Interface flow meter with net / forward / reverse totalisers, flow and a monitored 4-20mA loop (no pressure or meter alarms).

**Current values:** interfaceError, totaliser (net) / totaliserUnits, forwardTotaliser / forwardTotaliserUnits, reverseTotaliser / reverseTotaliserUnits, flow / flowUnits, loopReading / loopUnits, loopAlarm

**Datalog entry:** logDate, interfaceError, forwardTotaliser, reverseTotaliser, forwardConsumption, reverseConsumption, flow, loopReading, loopAlarm, batteryStatus, powerStatus, interpolatedEntry

# 6. getDeviceInformation Field Reference

## 6.1 Basic Device Fields (basicData=true)

| Field | Type | Description |
|---|---|---|
| updateConfiguration | Boolean | Device configuration update pending flag |
| dataIndex | Integer | Datalog table index key |
| zoneCode | String | Zone code |
| siteCode | String | Site code |
| deviceCode | String | Unique device code |
| commsStatus | String | Online / Offline / Sleep / Unknown |
| lastSeen | Date | Last time the device was seen (site local time) |
| timeZone | String | Site time zone |
| deviceType | String | Device type, e.g. CDS552 |
| deviceDescription | String | Device type description |
| applicationCode | String | Application code |
| applicationDescription | String | Application description |
| description | String | Device description |
| serialNumber | String | Serial number |
| gsmSignal | Integer | GSM signal strength in % |
| batteryStatus | Integer | Battery life remaining in % |
| powerStatus | String | Mains / Battery / Unknown |
| powerMode | String | Mains / Battery / Dual |
| alarm / alert | Boolean | Alarm and alert flags |
| alarmMessage / alertMessage | String | Alarm and alert messages |
| latitude / longitude | Double | Device position |
| wakeupPeriod | Integer | Wakeup period in minutes |
| datalogPeriod | Integer | Datalog period in minutes |
| lastDatalogDate | Date | Date / time of the last datalog entry (site local time) |
| lastFtpUpload | Date | Last FTP upload time (site local time) |
| userConfigurationData | String | User defined configuration data |

## 6.2 Full Device Fields (basicData=false)

The full response contains every basic field above plus every telemetry column, populated according to the device application (unused columns hold defaults):

| Field | Type | Description |
|---|---|---|
| interfaceError | Boolean | Interface error flag |
| totaliser / totaliserUnits | Double / String | Totaliser value and units |
| totaliser1 / totaliser1Units | Double / String | Totaliser 1 value and units |
| totaliser2 / totaliser2Units | Double / String | Totaliser 2 value and units |
| cumulativeTotaliser / cumulativeTotaliserUnits | Double / String | Cumulative totaliser value and units |

| | | |
|---|---|---|
| forwardTotaliser / forwardTotaliserUnits | Double / String | Forward totaliser value and units |
| reverseTotaliser / reverseTotaliserUnits | Double / String | Reverse totaliser value and units |
| counter1 / counter2 | Double | Krohne meter counters (m3) |
| pulse1Status / pulse2Status | Integer | Pulse input line status 0=Low 1=High |
| productVolume / productLevel | Double | Tank product volume (m3) and level (m) |
| tankCalibration | Double | Tank calibration value |
| cathodicVoltage / cathodicStatus | Double / String | Cathodic protection voltage and status (N/C/O) |
| loopReading / loopUnits / loopAlarm | Double / String / String | Loop value, units and alarm (None / High Alarm / Low Alarm / Unknown) |
| loop1Reading / loop1Units / loop1Alarm | Double / String / String | Loop 1 value, units and alarm |
| loop2Reading / loop2Units / loop2Alarm | Double / String / String | Loop 2 value, units and alarm |
| pressure / pressureUnits | Double / String | Pressure reading and units |
| pressureUpstream / pressureUpstreamUnits | Double / String | Upstream pressure reading and units |
| pressureDownstream / pressureDownstreamUnits | Double / String | Downstream pressure reading and units |
| flow / flowUnits | Double / String | Flow reading and units |
| medium | Integer | Medium through the meter (MBus code) |
| mediumTemperature / mediumTemperatureUnits | Double / String | Medium temperature and units |
| ambientTemperature / ambientTemperatureUnits | Double / String | Ambient temperature and units |
| input / output | Integer | Digital input / output status 0=Off 1=On |
| deviceAlarmData | String | Alarm data relating to the attached device |
| mbusStatus | Integer | MBus interface status byte |
| meterSerial | String | Meter serial number |
| meterMedium | String | Medium flowing through the meter (text) |
| meterBattery | Integer | Remaining days on meter battery |
| meterExternalVoltage / meterBatteryVoltage | Double | Meter external / battery voltages |
| meterExternalPowerStatus / meterInternalPowerStatus | String | Meter power status values |
| meterPowerSource | String | Power source for the meter |
| meterData / meterAlarmData | Integer | Meter manufacturer specific data / alarm data |
| meterFirmwareVersion | Decimal | Meter firmware version |
| meterSize | String | Size of the meter |
| soilTemperature / soilTemperatureUnits | Double / String | Soil temperature and units |
| soilMoisture / soilMoistureUnits | Double / String | Soil moisture and units |
| soilElectricalConductivity / soilElectricalConductivityUnits | Double / String | Soil electrical conductivity and units |
| soilPh | Double | Soil pH reading |
| soilNitrogen / soilNitrogenUnits | Double / String | Soil nitrogen and units |

| | | |
|---|---|---|
| soilPhosphorus / soilPhosphorusUnits | Double / String | Soil phosphorus and units |
| soilPotassium / soilPotassiumUnits | Double / String | Soil potassium and units |
| soilSensorManufactureDate | Date | Soil sensor manufacture date (pH expiry) |

# 7. Appendix — Value Reference

## 7.1 Status Values

| Field | Values |
|---|---|
| commsStatus / status | Online — device connected; Offline — mains device not connected; Sleep — battery device between wakeups; Unknown — not in the live register |
| powerStatus | Mains / Battery / Unknown (current power source) |
| powerMode | Mains / Battery / Dual (configured mode) |
| loopAlarm (and loop1/loop2) | None / High Alarm / Low Alarm / Unknown |
| wakeupType | Manual / Alarm / Auto (battery devices only) |
| cathodicStatus | N = None, C = Corroding, O = Overprotected |
| userType codes | N=Normal, S=Supervisor, O=Operator, T=Technician, A=Administrator, Z=Zone User, U=Site User, D=Device User |
| input / output | 0 = Off, 1 = On |
| pulse1Status / pulse2Status | 0 = Low, 1 = High |

## 7.2 Practical Notes

- Query dates for datalog calls are in the site's local time zone, not the caller's.
- The interpolatedEntry flag distinguishes real logged readings from values filled in by the hourly summariser.
- Battery devices are normally in Sleep state; they cannot receive sendRemoteCommand — use saveScriptCommand so the command is delivered at the next wakeup.
- deleteZone and deleteSite cascade to the devices beneath them; deleteDatalog permanently removes readings. There is no undo for any delete endpoint.
- Keep API keys secret; a key grants full read / write access to its entire database, including user management.