OPERATE. UNDERSTAND. VERIFY.

Integration API Developer Handbook

Uptime Control 02.15.1

Detailed procedures, clear responsibilities and traceable checks.

Edition 6 November 2026
Linux x86_64 + Docker | Browser application

No real customer data in illustrations. Usage rights follow the separate agreement.

Contents

01 The integration contract 3

02 Operator provisioning 3

03 Obtaining tokens and sending requests 4

04 Machine-day measurements: meaning and limits 4

05 Service events: timestamps and fault linkage 4

06 Machine fields 5

07 Service fields 6

08 Retries, concurrency and corrections 7

09 Development and acceptance plan 7

10 Run the reference client 8

01 The integration contract

Integration API v1 receives structured data. Your system sends records to Uptime; Uptime does not crawl third-party servers. Two distinct POST endpoints accept machine-day measurements and completed service events. Send exactly one JSON record per HTTP request. There is no batch or generic file-upload endpoint.

The supplied openapi.json is the machine-readable contract. Field names and case are exact. Unknown and duplicate fields are rejected. Maximum request size is 32,768 bytes and maximum JSON depth is four. Use HTTPS with certificate validation throughout.

The API is disabled by default. The operator explicitly enables it after technical-account provisioning, role assignment, machine allowlisting and acceptance testing. Personal accounts and browser cookies are not technical API credentials.

02 Operator provisioning

1. Create a confidential OpenID Connect client in the existing Keycloak realm with service accounts enabled. Enable client authentication; disable browser login and direct password grants for this technical client.

2. Record the client ID and immutable service-account user identifier. The latter must exactly match the eventual sub claim. Transfer the client secret through an approved secure channel only.

3. Add an audience mapper for uptime-api. The access token must contain this audience; tokens intended for another service are not accepted.

4. Add a hardcoded string claim uptime_integration with value true and include it in access tokens. The token must also carry azp equal to the approved client ID and sub equal to the service-account identifier.

5. Configure the required optional client scopes uptime.machine.write and/or uptime.service.write with Include in token scope enabled. Request only the scopes the sending system needs.

6. Register the external service account in Uptime using that exact external subject. Grant the established import permission for the intended machines, for example the responsible service-department role scoped to a machine. Do not assign global administration.

7. Configure Subject, ClientId, Enabled, Permissions and MachineIds under IntegrationApi. Subject and ClientId must both match. Permissions uses machine.write and/or service.write; MachineIds contains actual Uptime UUIDs, not serial numbers.

8. Enable IntegrationApi:Enabled and the specific client. Restart only the Uptime application in a controlled way, then check readiness, valid ingestion and rejection of an unrelated machine.

9. Document the configuration owner, revocation and backup procedure. Custom API configuration must be explicitly restored and verified during recovery; standard installer backup acceptance used the API-disabled configuration.

03 Obtaining tokens and sending requests

The operator supplies the real token URL; .invalid example addresses are deliberately unusable. Use OAuth2 client_credentials with application/x-www-form-urlencoded and the fields grant_type=client_credentials, client_id, client_secret and a space-separated scope list. Successful responses provide access_token, token_type and expires_in.

Cache tokens until shortly before expiration. Renew ahead of expiry instead of requesting a new token for every record. Never place tokens or secrets in source code, URLs, screenshots or logs. Obtain a new client-credentials token; do not assume a refresh token is available.

Set Content-Type: application/json and Authorization: Bearer <access_token>. The exact target paths are /api/integration/v1/machine-measurements and /api/integration/v1/service-events. Technical tokens are blocked outside this namespace, even when a client has been disabled; they cannot administer users.

On 401, obtain a fresh token once and retry the unchanged record. Stop and investigate a repeated 401. Do not loop on 403: check client enablement, scope, machine allowlist and the Uptime role assignment.

04 Machine-day measurements: meaning and limits

A record describes one machine on one local calendar day in the airport timezone. localDate uses yyyy-MM-dd. actualOperatingSeconds and relevantDowntimeSeconds are integer seconds, not Excel day fractions or labour minutes. reportedAvailabilityPercent is a JSON number from 0 to 100, not text containing a percent sign. Use a decimal point.

Operating duration cannot exceed that actual local day: normally 24 hours, potentially 23 or 25 at daylight-saving transitions. Downtime must be non-negative and no greater than operating duration. Reported availability must agree with operating duration and downtime within the implemented rounding tolerance. Do not infer a different calculation formula from the contract.

The machine must be active and in service on the given date. The extreme date 9999-12-31 is rejected because the next-day boundary cannot be calculated safely. Different existing values for the same machine day are never silently overwritten.

Your adapter translates source-system raw data into this fixed structure before transmission. Ingestion does not invent a contractual rule or turn a service report into a measured operating value.

05 Service events: timestamps and fault linkage

startUtc and endUtc are complete ISO-8601 timestamps with Z or an explicit offset. End must follow start. Accepted records describe completed actual work within the machine service life; end cannot be more than five minutes ahead of current server time. This endpoint is not a future maintenance planner.

Duration is derived from start and end. There is no total-time or technician-count field in this contract. Category and restoration status must exactly match documented enum values. Submit the fault description and action taken separately.

The stored machine serial must be normalizable to the SSL01-<number> format required by the underlying service importer. machineId nevertheless remains the UUID, not the serial number.

faultReference is optional. Reusing it for the same client and machine links the events to the same fault case. Your adapter must determine whether the event really continues or repeats that fault. Identical descriptive text does not establish this relationship. Different cases require different identifiers.

Subsequent availability calculations apply the effective Uptime rules. Do not subtract grace time in your adapter or artificially split events at midnight.

06 Machine fields

recordId

Required

JSON type: string

{"minLength": 1, "maxLength": 100, "pattern": "^[A-Za-z0-9_.-]+$"}

machineId

Required

JSON type: string

{"format": "uuid", "pattern":
"^[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}$"}

localDate

Required

JSON type: string

{"format": "date", "pattern": "^\\d{4}-\\d{2}-\\d{2}$"}

actualOperatingSeconds

Required

JSON type: integer

{"minimum": 0, "maximum": 108000}

relevantDowntimeSeconds

Required

JSON type: integer

{"minimum": 0, "maximum": 108000}

reportedAvailabilityPercent

Required

JSON type: number

{"minimum": 0, "maximum": 100}

07 Service fields

recordId

Required

JSON type: string

{"minLength": 1, "maxLength": 100, "pattern": "^[A-Za-z0-9_.-]+$"}

machineId

Required

JSON type: string

{"format": "uuid", "pattern":
"^[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}$"}

category

Required

JSON type: string

{"enum": ["MachineFault", "OperatorError", "PassengerInterference", "ThirdPartyFault",
"ScheduledMaintenance"]}

startUtc

Required

JSON type: string

{"format": "date-time", "maxLength": 40, "pattern":
"^\\d{4}-\\d{2}-\\d{2}T\\d{2}:\\d{2}:\\d{2}(\\.\\d{1,7})?(Z|[+-]\\d{2}:\\d{2})$"}

endUtc

Required

JSON type: string

{"format": "date-time", "maxLength": 40, "pattern":
"^\\d{4}-\\d{2}-\\d{2}T\\d{2}:\\d{2}:\\d{2}(\\.\\d{1,7})?(Z|[+-]\\d{2}:\\d{2})$"}

whatHappened

Required

JSON type: string

{"minLength": 1, "maxLength": 4000}

actionTaken

Required

JSON type: string

{"minLength": 1, "maxLength": 4000}

restorationStatus

Required

JSON type: string

{"enum": ["Restored", "ProvisionallyRestored", "StillOutOfService", "FollowUpRequired"]}

faultReference

Optional

JSON type: string

{"minLength": 1, "maxLength": 100, "pattern": "^[A-Za-z0-9_.-]+$"}

08 Retries, concurrency and corrections

Derive recordId from a stable source identifier and keep it unchanged across retries. Do not recycle it for another record within the same client. It accepts 1-100 characters from A-Z, a-z, 0-9, underscore, period and hyphen.

201/imported means newly stored. 200/already_present means identical content already exists; treat both as successful delivery. Persist recordId, receiptId, contentSha256 and delivery time in your own outbound journal. A receipt is not an official report release.

409/RECORD_CONFLICT indicates different existing content. Stop that record and investigate. Do not generate a new recordId merely to bypass conflict detection. Authorized service staff correct persisted service reports through the revision-controlled workflow; ingestion does not offer a general overwrite endpoint.

A connection failure does not prove storage failed. Retry identical content with the same identifier. For example, start after one second and double the delay with jitter up to an operational limit. Retry 429 and transient 5xx only within bounded policies; 400, 403, 409, 415 and 422 need investigation. A disabled ingestion endpoint can return 503.

Parallelize independent records only. Preserve ordering where machine-day or fault-case semantics require it. Multiple HTTP requests have neither a guaranteed global order nor a multi-record transaction.

09 Development and acceptance plan

1. Map source-system machine identifiers to approved Uptime UUIDs and obtain domain-owner verification.

2. Define stable source identifiers per record type. Persist each record in a durable outbound queue before transmission.

3. Implement field mapping, timezone/offset handling, numeric validation and correct fault-case linkage. Detect incomplete or contradictory source data before sending.

4. Implement token acquisition, secret storage, timeouts, bounded retry and credential-free logging.

5. Submit a valid machine day and service event: expect 201. Retry identical content: expect 200. Change a valid value using the same identifier: expect 409.

6. Test wrong scope, unrelated machine and disabled client: expect 403. Invalid bearer tokens must receive 401. A completely missing token can be rejected earlier with 400 by browser protection; no record may be stored.

7. Test unknown fields, wrong case, malformed JSON, invalid intervals, daylight-saving day lengths and the 32,768-byte limit. Retain input, observed result and expected status.

8. Inspect persisted records and provenance in Uptime. Test connection loss, concurrent duplicate requests and outbound-queue recovery.

9. Agree ownership, incident handling, maintenance and revocation. Enable production ingestion only after documented acceptance for the intended machines.

10 Run the reference client

client.py is a readable Python 3 integration example without additional packages. It belongs to the API documentation, not the Uptime application source.

1. Check the synthetic examples locally:

```
python3 client.py machine machine-example.json --validate-only
```

```
python3 client.py service service-example.json --validate-only
```

This check covers field names, required fields, JSON structure and size only. Business values, permissions and acceptance are validated by the server.

2. Supply UPTIME_URL, UPTIME_TOKEN_URL, UPTIME_CLIENT_ID and UPTIME_CLIENT_SECRET through secure runtime configuration. UPTIME_CA_FILE optionally selects a trusted private CA. Do not commit credentials or put them in shared terminal history.

3. Send in your approved test environment:

```
python3 client.py machine machine-example.json --receipt machine-receipt.json
```

```
python3 client.py service service-example.json --receipt service-receipt.json
```

The example uses HTTPS, rejects automatic redirects and retries transient failures a bounded number of times with the same recordId. It does not implement a durable outbound queue. Add monitoring, persistent queuing, receipt reconciliation and the acceptance tests described here to your application. A local STRUCTURE_OK does not mean server acceptance. Adapt example values and IDs to your synthetic test machine.

Access after installation

Your administrator activates Uptime Control once for the server. All users then sign in with their individual accounts. One purchase code covers the installation.



If Uptime Control aktivieren appears, contact your local administrator. Activation requires the purchase code issued for this installation. Creating another user account will not activate the server.

See 00_Installation/Aktivierung_EN.pdf for illustrated instructions. After activation follow the workflows and permissions in your role manual.