ANWENDEN. VERSTEHEN. NACHVOLLZIEHEN.

API-Schnittstellen Entwicklerhandbuch

Uptime Control 02.15.1

Detaillierte Abläufe, klare Zuständigkeiten und nachvollziehbare Kontrollen.

Ausgabe 06. November 2026
Linux x86_64 + Docker | Browseranwendung

Keine echten Kundendaten in Abbildungen. Nutzungsrechte richten sich nach der gesonderten Vereinbarung.

Inhalt

01 Verbindlicher Integrationsvertrag 3

02 Einrichtung durch den Betreiber 3

03 Tokenabruf und HTTP-Anfrage 4

04 Maschinen-Tageswerte: Bedeutung und Grenzen 4

05 Serviceereignisse: Zeiten und Fehlerzuordnung 4

06 Maschinenfelder 5

07 Servicefelder 6

08 Wiederholungen, Parallelität und Korrekturen 7

09 Entwicklungs- und Abnahmeplan 7

10 Referenzclient ausführen 8

01 Verbindlicher Integrationsvertrag

Integration API v1 nimmt strukturierte Daten entgegen. Ihr System sendet an Uptime; Uptime durchsucht nicht eigenständig fremde Server. Eine Verbindung besteht aus zwei getrennten POST-Endpunkten für Maschinen-Tageswerte und abgeschlossene Serviceereignisse. Pro HTTP-Aufruf wird genau ein JSON-Datensatz übertragen. Es gibt keinen Batch-Upload und keinen allgemeinen Datei-Upload über diese Schnittstelle.

Die mitgelieferte openapi.json ist die maschinenlesbare Spezifikation. Feldnamen und Groß-/Kleinschreibung sind verbindlich. Unbekannte oder doppelte Felder werden zurückgewiesen. Maximal 32.768 Bytes je Anfrage, maximale JSON-Verschachtelung vier. Ausschließlich HTTPS mit gültiger Zertifikatsprüfung verwenden.

Die API ist standardmäßig ausgeschaltet. Nach Einrichtung des technischen Kontos, Rechtezuweisung, Anlagenfreigabe und bestandener Integrationsabnahme aktiviert der Betreiber sie ausdrücklich. Persönliche Benutzerkonten und Sitzungs-Cookies sind keine technischen API-Zugangsdaten.

02 Einrichtung durch den Betreiber

1. Erstellen Sie im bestehenden Keycloak-Realm einen vertraulichen OpenID-Connect-Client mit aktivem Servicekonto. Client-Authentifizierung aktivieren; Browser-Login und direkte Passwortanmeldung für diesen technischen Client deaktivieren.

2. Notieren Sie Client-ID und die unveränderliche Benutzerkennung des Servicekontos. Die Benutzerkennung muss später exakt dem sub-Claim entsprechen. Das Client-Geheimnis nur über einen geschützten Kanal an den Betreiber der sendenden Anwendung geben.

3. Erstellen Sie den Audience-Mapper für uptime-api. Der Access-Token muss diese Zielgruppe enthalten; ein für einen anderen Dienst bestimmter Token wird nicht akzeptiert.

4. Erstellen Sie einen festen String-Claim uptime_integration mit dem Wert true. Aktivieren Sie ihn für Access-Tokens. Der technische Token muss außerdem azp mit der vereinbarten Client-ID und sub mit der Servicekonto-Kennung enthalten.

5. Richten Sie die benötigten optionalen Client-Scopes uptime.machine.write und/oder uptime.service.write ein. Include in token scope muss aktiv sein. Fordern Sie beim Tokenabruf nur tatsächlich benötigte Scopes an.

6. Registrieren Sie das externe Servicekonto in der Uptime-Benutzerverwaltung mit exakt dieser externen Kennung. Weisen Sie die vorhandene Importberechtigung für die vorgesehenen Anlagen zu, beispielsweise über die Rolle des zuständigen Servicebereichs mit Anlagenbezug. Keine globale Administratorrolle vergeben.

7. Tragen Sie Subject, ClientId, Enabled, Permissions und MachineIds in die IntegrationApi-Konfiguration ein. Subject und ClientId müssen gemeinsam passen. Permissions enthält machine.write beziehungsweise service.write; MachineIds enthält die tatsächlichen Uptime-UUIDs, nicht Seriennummern.

8. Aktivieren Sie IntegrationApi:Enabled und den konkreten Client. Starten Sie nur die Uptime-Anwendung kontrolliert neu und prüfen Sie Bereitschaft, gültigen Import und abgelehnte Fremdanlage.

9. Dokumentieren Sie Konfiguration, Verantwortliche, Widerruf und Sicherungsweg. Individuelle API-Konfiguration muss beim Restore ausdrücklich wiederhergestellt und geprüft werden; das Standard-Installationsbackup wurde mit deaktivierter API abgenommen.

03 Tokenabruf und HTTP-Anfrage

Die tatsächliche Token-URL nennt der Betreiber; die .invalid-Adressen in Beispielen sind absichtlich nicht erreichbar. Verwenden Sie OAuth2 client_credentials mit application/x-www-form-urlencoded. Felder: grant_type=client_credentials, client_id, client_secret und scope als durch Leerzeichen getrennte Liste. Ein erfolgreiches Ergebnis enthält access_token, token_type und expires_in.

Access-Tokens bis kurz vor Ablauf zwischenspeichern. Vor Ablauf erneuern, nicht für jeden Datensatz neu anmelden. Token und Geheimnisse niemals in Quellcode, URLs, Screenshots oder Protokolle schreiben. Ein Client-Credentials-Token wird durch erneuten Tokenabruf ersetzt; einen vorhandenen Refresh-Token nicht voraussetzen.

Für Uptime-Anfragen Content-Type: application/json und Authorization: Bearer <access_token> setzen. Der Zielpfad ist exakt /api/integration/v1/machine-measurements oder /api/integration/v1/service-events. Technische Tokens sind außerhalb dieses Bereichs gesperrt; auch bei deaktiviertem Client sind sie kein Weg in die Benutzerverwaltung.

Bei 401 einen Token einmal neu beziehen und den unveränderten Datensatz wiederholen. Bei erneutem 401 Konfiguration prüfen und Verarbeitung anhalten. Bei 403 keine Endlosschleife starten: Clientfreigabe, Scope, Anlagenliste und Uptime-Zuweisung prüfen.

04 Maschinen-Tageswerte: Bedeutung und Grenzen

Ein Datensatz beschreibt eine Anlage an einem lokalen Kalendertag der Flughafen-Zeitzone. localDate ist yyyy-MM-dd. actualOperatingSeconds und relevantDowntimeSeconds sind ganze Sekunden, keine Excel-Tagesbruchteile oder Personenminuten. reportedAvailabilityPercent ist ein JSON-Zahlenwert zwischen 0 und 100, kein Text mit Prozentzeichen. Dezimalpunkt verwenden.

Die Betriebsdauer darf die tatsächliche Länge dieses lokalen Tages nicht überschreiten: regulär 24 Stunden, am Sommerzeitwechsel gegebenenfalls 23 oder 25. Downtime darf weder negativ noch größer als die Betriebsdauer sein. Der gemeldete Prozentwert muss innerhalb der implementierten Rundungstoleranz mit Betriebsdauer und Downtime vereinbar sein. Aus der Spezifikation keine abweichende Rechenmethode ableiten.

Die Anlage muss aktiv und für das Datum in Betrieb sein. Der Grenzwert 9999-12-31 wird verworfen, weil kein Folgetag zur sicheren Tagesgrenze berechnet werden kann. Bereits vorhandene andere Werte für denselben Anlagentag werden nicht still überschrieben.

Vor dem Senden muss Ihr Adapter die eigenen Rohdaten in diese Struktur übersetzen. Die Integration erfindet keine Vertragsregel und setzt keinen Servicebericht gleich einem Messwert.

05 Serviceereignisse: Zeiten und Fehlerzuordnung

startUtc und endUtc sind vollständige ISO-8601-Zeitpunkte mit Z oder eindeutigem Offset. Ende muss nach Beginn liegen. Zulässig sind abgeschlossene tatsächliche Intervalle innerhalb des Anlagenlebens; Ende darf nicht mehr als fünf Minuten nach der aktuellen Serverzeit liegen. Die Schnittstelle ist kein zukünftiger Wartungsplaner.

Die Dauer wird aus Beginn und Ende abgeleitet. Ein Feld Gesamtzeit oder Technikeranzahl existiert in diesem Vertrag nicht. Kategorie und Wiederherstellungszustand müssen exakt aus den zulässigen Enum-Werten gewählt werden. Fehlersituation und Maßnahme werden getrennt als Text übermittelt.

Die Bestands-Seriennummer muss für den zugrunde liegenden Serviceimport in das geforderte SSL01-<Nummer>-Format normalisierbar sein. machineId ist trotzdem die UUID; sie wird nicht durch die Seriennummer ersetzt.

faultReference ist optional. Wird dieselbe Kennung beim selben Client und derselben Anlage erneut verwendet, verbindet sie die betreffenden Ereignisse fachlich zu einem Fehlerfall. Deshalb muss Ihre Anwendung entscheiden, ob ein Vorgang wirklich dieselbe Ursache/Fortsetzung ist. Gleicher Beschreibungstext bewirkt keine automatische Gleichsetzung. Unterschiedliche Fehlerfälle müssen unterschiedliche Kennungen erhalten.

Die spätere Verfügbarkeitsberechnung nutzt die wirksamen Uptime-Regeln. Ihr Adapter darf weder die Freifrist von der Dauer abziehen noch Ereignisse um Mitternacht künstlich aufteilen.

06 Maschinenfelder

recordId

Pflichtfeld

JSON-Typ: string

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

machineId

Pflichtfeld

JSON-Typ: 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

Pflichtfeld

JSON-Typ: string

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

actualOperatingSeconds

Pflichtfeld

JSON-Typ: integer

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

relevantDowntimeSeconds

Pflichtfeld

JSON-Typ: integer

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

reportedAvailabilityPercent

Pflichtfeld

JSON-Typ: number

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

07 Servicefelder

recordId

Pflichtfeld

JSON-Typ: string

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

machineId

Pflichtfeld

JSON-Typ: 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

Pflichtfeld

JSON-Typ: string

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

startUtc

Pflichtfeld

JSON-Typ: 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

Pflichtfeld

JSON-Typ: 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

Pflichtfeld

JSON-Typ: string

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

actionTaken

Pflichtfeld

JSON-Typ: string

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

restorationStatus

Pflichtfeld

JSON-Typ: string

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

faultReference

Optional

JSON-Typ: string

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

08 Wiederholungen, Parallelität und Korrekturen

recordId muss aus einer stabilen Quellkennung entstehen und bei jedem Wiederholungsversuch unverändert bleiben. Sie darf pro Client nicht für einen anderen Datensatz recycelt werden. Zulässig sind 1 bis 100 Zeichen aus A-Z, a-z, 0-9, Unterstrich, Punkt und Bindestrich.

201/imported bedeutet neu gespeichert. 200/already_present bedeutet identischer Inhalt bereits vorhanden; behandeln Sie beides als erfolgreiche Zustellung. Speichern Sie recordId, receiptId, contentSha256 und Zustellzeit in Ihrem eigenen Versandjournal. Die Antwort ist keine offizielle Berichtsfreigabe.

409/RECORD_CONFLICT bedeutet abweichender bereits vorhandener Inhalt. Stoppen Sie den betroffenen Datensatz und klären Sie die Ursache. Keine neue recordId erzeugen, nur um die Konfliktprüfung zu umgehen. Ein gespeicherter Servicebericht wird durch den berechtigten Servicebereich revisionssicher korrigiert; die Ingestion-API bietet dafür keinen allgemeinen Überschreibendpunkt.

Bei Netzwerkabbruch ist unklar, ob die Speicherung bereits erfolgt ist. Wiederholen Sie denselben Inhalt mit derselben Kennung. Beginnen Sie beispielsweise nach einer Sekunde und verdoppeln Sie die Wartezeit mit Zufallsanteil bis zu einer betrieblich festgelegten Grenze. 429 und vorübergehende 5xx dürfen begrenzt wiederholt werden; 400, 403, 409, 415 und 422 verlangen Prüfung. Ein ausgeschalteter Eingang kann 503 zurückgeben.

Parallele Verarbeitung nur für unabhängige Datensätze verwenden. Pro Anlage und Tag beziehungsweise pro fachlichem Fehlerfall die Reihenfolge kontrollieren. Es gibt keine zugesicherte globale Reihenfolge mehrerer HTTP-Aufrufe und keine Mehrdatensatz-Transaktion.

09 Entwicklungs- und Abnahmeplan

1. Erstellen Sie eine Zuordnungstabelle Ihrer Anlagenkennungen zu den freigegebenen Uptime-UUIDs. Lassen Sie diese fachlich prüfen.

2. Legen Sie pro Datentyp die stabile Quellkennung fest. Schreiben Sie den Datensatz vor dem Versand in eine dauerhafte lokale Ausgangswarteschlange.

3. Implementieren Sie Feldmapping, Zeitzonen-/Offsetumrechnung, numerische Prüfung und die ausdrücklich richtige Fehlerzuordnung. Prüfen Sie fehlende und widersprüchliche Quelldaten vor dem Versand.

4. Implementieren Sie Tokenabruf, geschützte Geheimnisablage, Timeout, begrenzte Wiederholung und Protokollierung ohne Zugangsdaten.

5. Prüfen Sie einen gültigen Maschinentag und Serviceeintrag: erwartet 201. Wiederholen Sie identisch: erwartet 200. Ändern Sie einen fachlich gültigen Wert mit derselben Kennung: erwartet 409.

6. Prüfen Sie falschen Scope, fremde Anlage und deaktivierten Client: erwartet 403. Ein ungültiger Bearer-Token muss 401 erhalten. Ohne Token kann der vorgeschaltete Browserschutz bereits 400 erzeugen; in keinem Fall darf gespeichert werden.

7. Prüfen Sie unbekannte Felder, falsche Schreibweise, beschädigtes JSON, unzulässige Zeitintervalle, Sommerzeit-Tageslängen und 32.768-Byte-Grenze. Bewahren Sie Eingabe, Ergebnis und erwarteten Status auf.

8. Kontrollieren Sie in Uptime den gespeicherten Eintrag und seine Herkunft. Testen Sie Verbindungsabbruch, parallele Wiederholung und Wiederanlauf Ihrer Warteschlange.

9. Vereinbaren Sie Verantwortliche, Meldungsweg, Wartungsfenster und Widerruf. Aktivieren Sie die Produktionsanbindung erst nach dokumentierter Abnahme mit ausschließlich vorgesehenen Anlagen.

10 Referenzclient ausführen

Die Datei client.py ist ein lesbares Integrationsbeispiel für Python 3 ohne Zusatzpakete. Sie gehört zur API-Dokumentation und ist kein Quellcode der Uptime-Anwendung.

1. Prüfen Sie zunächst die synthetischen Beispieldateien lokal:

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

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

Diese Prüfung kontrolliert nur Feldnamen, Pflichtfelder, JSON-Struktur und Größe. Fachwerte, Rechte und Annahme prüft anschließend der Server.

2. Setzen Sie UPTIME_URL, UPTIME_TOKEN_URL, UPTIME_CLIENT_ID und UPTIME_CLIENT_SECRET über Ihre sichere Laufzeitkonfiguration. Optional UPTIME_CA_FILE für eine eigene vertrauenswürdige CA. Zugangsdaten nicht in Dateien im Repository oder in gemeinsam einsehbare Terminalverläufe schreiben.

3. In Ihrer freigegebenen Testumgebung senden:

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

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

Das Beispiel verwendet HTTPS, verhindert automatische Weiterleitungen und wiederholt vorübergehende Fehler begrenzt mit unveränderter recordId. Es implementiert keine dauerhafte Ausgangswarteschlange. Ergänzen Sie für Ihre eigene Anwendung Überwachung, persistente Warteschlange, Quittungsvergleich und die Abnahmetests aus diesem Handbuch. Ein lokales STRUCTURE_OK bedeutet keine erfolgreiche Übernahme. Die Beispieldaten und IDs müssen auf Ihre synthetische Testanlage angepasst werden.

Zugang nach der Installation

Ihr Administrator aktiviert Uptime Control einmal für den Server. Alle Benutzer melden sich anschließend mit ihrem persönlichen Konto an. Es gibt keinen Kaufcode pro Benutzer.



Erscheint beim Aufruf „Uptime Control aktivieren“, wenden Sie sich an Ihren lokalen Administrator. Er benötigt den für diese Installation ausgestellten Kaufcode. Eine neue Benutzerregistrierung löst die Lizenzanforderung nicht.

Die bebilderte Anleitung liegt unter 00_Installation/Aktivierung_DE.pdf. Nach der Freischaltung gelten die im Rollenhandbuch beschriebenen Arbeitsabläufe und Berechtigungen.