Externes API

GATEWatch verfügt über ein externes API zum Auslesen und Hinzufügen, sowie Ändern von Gästen.

- Übersicht
- Abruf der Metadaten (metadata)
- Abruf von Gast-Daten als Liste (pull)
- Abruf von Daten eines einzelnen Gastes (fetch)
- Hinzufügen und Aktualisieren von Gast-Daten (push)

Übersicht

Zugang zur API

Für die Nutzung der API ist mindestens ein gültiger **API-Schlüssel** erforderlich (Hauptschlüssel). Dieser kann über das zuständige Teilnehmermanagement bezogen werden. Abhängig vom Anwendungsfall können zusätzlich weitere Schlüssel zur Verfügung gestellt werden. Die Verwendung und Funktion der einzelnen Schlüssel wird im weiteren Verlauf dieser Dokumentation erläutert. Kann die Authentifizierung nicht erfolgreich durchgeführt werden kommt es zu einem HTTP 401 Fehler.

Zusätzlich kann der Zugriff auf die API über eine **IP-Filterung** eingeschränkt werden. Es wird empfohlen, die IP-Adresse(n) der anfragenden Server (inkl. Testumgebungen) im Vorfeld zu definieren und bereitzuhalten. Schlägt die Prüfung der IP Adresse fehl, kommt es zu einem HTTP 401 Fehler.

Die API verwendet **JSON (JavaScript Object Notation)** als Austauschformat. Anfragen an die API sowie die von ihr zurückgegebenen Antworten müssen im JSON-Format strukturiert bzw. verarbeitet werden.

API-Endpunkte

Die API bietet drei dedizierte Endpunkte zur Interaktion:

- `https://gwatch.events/ext-api/metadata`
  Zweck: Abfrage verfügbarer Metadaten und Konfigurationsinformationen.
- `https://gwatch.events/ext-api/pull`
  Zweck: Abruf vorhandener Gast-Daten.
- `https://gwatch.events/ext-api/push`
  Zweck: Übertragung (Hinzufügen oder Aktualisieren) von Gast-Daten.

Weitere Informationen

Die genaue Struktur der Anfragen, das erwartete JSON-Format sowie mögliche Fehlermeldungen und Antwort-Codes werden auf den folgenden Seiten ausführlich beschrieben.

Falls Sie die API online testen möchten, besuchen Sie:

https://gwatch.events/ext-api

Abruf der Metadaten (metadata)

Aufruf

Der Abruf der Metadaten erfolgt über eine **HTTP-GET-Anfrage** an folgende URL:

`https://gwatch.events/ext-api/metadata`

Zur Authentifizierung muss der **API-Hauptschlüssel** im Request-Header unter dem Schlüssel `X-API-key` mitgesendet werden. Kann die Authentifizierung nicht erfolgreich durchgeführt werden kommt es zu einem HTTP 401 Fehler.

Struktur der Antwort

Die Antwort der API ist in drei logische Bereiche gegliedert:

1. Veranstaltungsinformationen

Dieser Abschnitt enthält allgemeine Informationen zur Veranstaltung. Besonders wichtig ist hier der Eintrag zur **Standardlänge des Ticketcodes**. Dieser Wert ist beim Hinzufügen neuer Gäste relevant, um sicherzustellen, dass keine Konflikte mit bestehenden Codes entstehen. Es kann ein längerer oder kürzerer Code verwendet werden, sofern er eindeutig ist.

Zusätzlich werden die in der Veranstaltung verfügbaren **Sprachen** mitgeliefert. Diese Informationen dienen dazu, beim Erstellen oder Aktualisieren von Gästen die passende Sprache zu wählen.

Beispiel:

```
"event": {
        "name": "IT-Seminar",
        "date": "2025-05-31",
        "defaultTicketCodeLength": 10,
        "locales": [
            "de", "en"
        ]
    },
```

2. Verfügbare Felder für Gäste

In diesem Abschnitt werden alle Felder aufgeführt, die beim Hinzufügen oder Bearbeiten von Gästen verwendet werden können. Enthält ein Feld spezifische **Restriktionen**, so sind diese im jeweiligen Feld mit angegeben. Wenn in den Restriktionen `"nullable": false` angegeben ist, bedeutet dies, dass das Feld beim Einfügen oder Bearbeiten nicht mit einem Nullwert übergeben werden darf. Sind für das Feld keine Daten vorhanden, kann es aber einfach weggelassen werden.

Jedes Feld enthält mindestens folgende Informationen:

- `key`: Hier findet sich der Schlüssel des Feldes, welcher zum Hinzufügen und Bearbeiten genutzt werden muss.
- `name`: Dieser Name dient zur Kommunkation mit dem Teilnehmermanagement, damit alle wissen um welches Feld es sich handelt.
- `type`: Der Typ des Feldes, welche bestimmte Validierungsregeln impliziert.
  - `multiLineText`: Text, der Zeilenumbrüche enthalten darf.
  - `singleLineText`: Text, der keine Zeilenumbrüche enthalten darf.
  - `email`: Der Inhalt muss eine formal gültige E-Mail-Adresse sein.
  - `url`: Der Inhalt muss eine formal gültige URL sein.
  - `numeric`: Der Inhalt muss eine formal gültige Zahl im "Decimal Point Notation"-Format sein.
  - `integer`: Der Inhalt muss eine formal gültige Ganzzahl sein.
  - `boolean`: Der Inhalt kann "Yes", "No", 1, 0, true oder false sein.
  - `date`: Format muss `YYYY-MM-DD` sein, z. B. `2025-04-08`.
  - `dateTime`: Format muss `YYYY-MM-DDTHH:MM:SS` sein, z. B. `2025-04-08T14:38:14`. Zeitzonenangaben werden nicht verwendet.
  - `list`: Die möglichen Optionen werden als Key-Value-Paare angegeben. Beim Hinzufügen oder Bearbeiten von Gästen ist der **Key** zu verwenden, bei der Abfrage wird die **lokalisierte Value** zurückgegeben (abhängig von der gewählten Sprache).

Beispiel:

```
"guest": {
      "key": "GUEST_TICKET_CODE",
      "name": "Ticket code",
      "type": "singleLineText",
      "restrictions": {
          "nullable": false
          "unique": true,
          "min": 4,
          "max": 128
      }
  },
```

```
  {
      "key": "GUEST_LOCALE",
      "name": "Locale",
      "type": "list",
      "items": [
          {
              "key": "de",
              "value": "Deutsch"
          }
      ],
      "restrictions": {
          "nullable": false
      }
  },
  {
      "key": "GUEST_DINNER",
      "name": "Teilnahme am Abendessen",
      "type": "boolean"
  },
  . . . .
}
```

3. Gates (Zugangsbereiche)

Der dritte Bereich enthält Informationen zu sogenannten **Gates**. Diese abstrakten Einheiten können verschiedene Bedeutung haben – zum Beispiel Veranstaltungstage, Sessions, Vorträge oder anderweitig geschützte Bereiche.

Diese Informationen dienen dazu, Gästen beim Erstellen oder Bearbeiten bestimmte Gates zuzuordnen. Eine detaillierte Verwendung wird im Abschnitt zum `pull`-Aufruf erläutert.

Beispiel:

```
"gates: [
    {
        "id": 6398,
        "name": "Sicherheit von Websites"
    },
    {
```

```
        "id": 6399,
        "name": "Anti-Viren Software aktuell"
    },
    . . . .
  ]
}
```

Abruf von Gast-Daten als Liste (pull)

Aufruf

Der Abruf der Gast-Daten als Liste erfolgt über eine **HTTP-GET-Anfrage** an folgende URL:

`https://gwatch.events/ext-api/pull`

Zur Authentifizierung ist im Request-Header der **API-Hauptschlüssel** mit dem Schlüssel `X-Api-key` zu übergeben. Kann die Authentifizierung nicht erfolgreich durchgeführt werden kommt es zu einem HTTP 401 Fehler.

Varianten / Datenansichten

Die zurückgegebenen Inhalte können serverseitig individuell konfiguriert werden. Falls mehrere Varianten (sogenannte **Flavors**) existieren, erhalten Sie für jede Variante einen separaten Schlüssel, der im Header als `X-Api-flavor` übergeben werden muss. Kann die Authentifizierung nicht erfolgreich durchgeführt werden kommt es zu einem HTTP 401 Fehler.

Für jede Variante kann optional ein eigener **IP-Filter** definiert sein. Schlägt die Prüfung der IP Adresse fehl, kommt es zu einem HTTP 401 Fehler.

Optionale GET-Parameter

Folgende Parameter können zusätzlich in der URL übergeben werden:

- **limit**
  Die API liefert pro Aufruf maximal **2.500 Datensätze** zurück.
  Bei komplexen internen Berechnungen (z. B. bei kombinierten Feldwerten) kann eine Reduktion des Limits hilfreich sein, um Timeouts zu vermeiden.
- **offset**
  Um mehr als `limit` Datensätze abzufragen, kann ein Offset-Wert verwendet werden.
  Fehlt dieser, wird automatisch `0` angenommen.
  Beispielhafte Paginierung:
  - 1. Aufruf: `limit=100`

    - 2. Aufruf: `limit=100&offset=100`
    - 3. Aufruf: `limit=100&offset=200`
    - usw.
- **changesUntil**
  Zur effizienten Synchronisation Ihres Datenbestands können Sie mit diesem Parameter nur Datensätze abrufen, die **seit einem bestimmten Zeitpunkt geändert wurden**.
  Format: `YYYY-MM-DDTHH:MM:SS`
  Beispiel: `changesUntil=2025-04-08T15:00:00`

Zusatzinformationen je Datensatz

Unabhängig von den für Sie konfigurierten Feldern enthält jeder zurückgegebene Datensatz folgende Zusatzinformationen:

- **ID**: Die eindeutige Datenbank-ID des Gastes.
- **created**: Zeitstempel der Erstellung des Datensatzes.
- **updated**: Zeitstempel der letzten Änderung.
- **deleted**: `true` oder `false`.
  Wurde ein Gast zwischenzeitlich gelöscht, wird dies über dieses Feld signalisiert. So können Sie auch in Ihrem System entsprechende Einträge entfernen. In diesem Fall sind alle konfigurierten Felder im Datensatz **leer**.

Lokalisierte Darstellung

Die Ausgabe der Daten erfolgt für jeden Gast **lokalisiert**, das heißt:

- Datumsangaben, Zeitstempel und Zahlen mit Dezimalstellen werden entsprechend der Sprache des Gastes formatiert.
- Listenfelder enthalten als Wert die **lokalisierte** Bezeichnung im Kontext der Sprache des Gastes (nicht den Schlüssel).

Dies gewährleistet eine benutzerfreundliche Darstellung der Daten im jeweiligen Sprachkontext.

Fehlermeldungen und Rückgabeformat

Im Fehlerfall gibt die API eine strukturierte Rückmeldung und den HTTP Fehler 422 zurück. Das Feld `error` ist in diesem Fall auf `true` gesetzt. Das Feld `message` enthält die genaue Beschreibung des Fehlers.

Beispiel (Fehlerhafte Anfrage):

```
{
    "error": true,
    "message": {
        "limit": [
            "The limit must be at least 1."
        ]
    }
}
```


Im Erfolgsfall ist `error` auf `false` gesetzt. Die Antwortstruktur gliedert sich in:

- `meta`: Enthält Informationen zu den verwendeten Parametern und verfügbaren Datensätzen.
- `event`: Liste der konfigurierten Felder zur Veranstaltung.
- `guests`: Liste der gefundenen Gast-Datensätze. Bis auf die letzten vier Schlüssel `id`, `created`, `updated` und `deleted` können alle Schlüssel frei vereinbart werden. Sie entsprechen nicht zwingend den Schlüsselwerten des `/metadata` Aufrufs, da die Inhalt hier aus verschiedenen Quellen kombiniert und z.T. auch berechnet werden können.

Beispiel (Erfolgreiche Antwort):


```
{
    "error": false,
    "meta": {
        "changesUntil": "2025-02-23T15:00:00",
        "available": 19,
        "limit": 1,
        "offset": 0,
        "nextUrl": "https://gwatch.events/ext-api/pull?limit=1&offset=1&changesUntil=2025-02-23T15:00:00"
    },
    "event": [
        "name": "IT-Forum"
    ],
    "guests": [
        {
            "Vorname": "Susi",
            "Nachname": "Sorglos",
            "Mitgliedschaft": "IT-Security Concepts",
            "Letzter Scan": "20.03.2025 19:57:57",
            "id": 347639,
```

```
            "created": "2017-01-07T20:08:30",
            "updated": "2025-03-20T20:13:47",
            "deleted": false
        }
        . . . .
    ]
}
```

Abruf von Daten eines einzelnen Gastes (fetch)

Aufruf

Der Abruf der Gast-Daten erfolgt über eine **HTTP-GET-Anfrage** an folgende URL:

`https://gwatch.events/ext-api/fetch?code=<Ticketcode>`

Zur Authentifizierung ist im Request-Header der **API-Hauptschlüssel** mit dem Schlüssel `X-Api-key` zu übergeben. Kann die Authentifizierung nicht erfolgreich durchgeführt werden kommt es zu einem HTTP 401 Fehler.

Varianten / Datenansichten

Die zurückgegebenen Inhalte können serverseitig individuell konfiguriert werden. Falls mehrere Varianten (sogenannte **Flavors**) existieren, erhalten Sie für jede Variante einen separaten Schlüssel, der im Header als `X-Api-flavor` übergeben werden muss. Kann die Authentifizierung nicht erfolgreich durchgeführt werden kommt es zu einem HTTP 401 Fehler.

Für jede Variante kann optional ein eigener **IP-Filter** definiert sein. Schlägt die Prüfung der IP Adresse fehl, kommt es zu einem HTTP 401 Fehler.

Notwendiger GET-Parameter

Folgender Parameter muss zusätzlich in der URL übergeben werden:

- **code:** Dies ist der Ticketcode des Gastes, dessen Daten erfragt werden sollen.

Lokalisierte Darstellung

Die Ausgabe der Daten erfolgt für jeden Gast **lokalisiert**, das heißt:

- Datumsangaben, Zeitstempel und Zahlen mit Dezimalstellen werden entsprechend der Sprache des Gastes formatiert.

- Listenfelder enthalten als Wert die **lokalisierte** Bezeichnung im Kontext der Sprache des Gastes (nicht den Schlüssel).

Dies gewährleistet eine benutzerfreundliche Darstellung der Daten im jeweiligen Sprachkontext.

Statusmeldungen und Rückgabeformat

- **401**: nicht autorisiert. Dies bedeutet, dass entweder der API Key und/oder der API Flavor key nicht korrekt sind.
- **403**: unter dem übergebenen Code konnte ein Gast gefunden werden. Dieser hat aber der Weitergabe seiner Daten nicht zugestimmt.
- **404**: unter dem übergebenen Code konnte kein Gast gefunden werden.
- **422**: der notwenige Parameter (code) fehlt.
- **200**: erfolgreiche Antwort

Beispiel (Erfolgreiche Antwort):

```
{
    "Vorname": "Susi",
    "Nachname": "Sorglos",
    "Mitgliedschaft": "IT-Security Concepts"
}
```

Hinzufügen und Aktualisieren von Gast-Daten (push)

Aufruf

Die Übertragung erfolgt über eine **HTTP-POST-Anfrage** an folgende URL:

`https://gwatch.events/ext-api/push`

Authentifizierung

Folgende Header müssen gesetzt werden:

- `X-Api-key`: Haupt-API-Schlüssel (Pflicht)
- `X-Api-push`: zusätzlicher Import-Schlüssel (optional, empfohlen – bitte separat anfordern)

Request-Body (Payload)

1. meta

Enthält Steuerparameter für das Importverhalten:

- **failIfExists** (Boolean): Gibt an, ob bei vorhandenem Ticketcode ein Fehler erzeugt wird oder die Daten aktualisiert werden.
- **insertAs** (Integer): Definiert den Status für neue Datensätze:
  - `1`: bereits bestätigt
  - `2`: zu bestätigen (erfordert spätere Rückmeldung durch den Gast)

  Hinweis: Wird nur bei neu angelegten Gästen berücksichtigt.

2. guests

Enthält ein Array mit bis zu **100 Gast-Datensätzen**. Jeder Datensatz ist ein Key-Value-Objekt, basierend auf den Feldern aus `/metadata`.

Besonderheiten:

- **Ticketcode**: Fehlt er, wird ein neuer generiert. Ist er vorhanden, erfolgt eine Prüfung – abhängig von `failIfExists`.
- **Gates**: Falls im Metadata-Abschnitt `gates` enthalten sind, kann zusätzlich ein Array `gates` mit Gate-IDs gesetzt werden. **Ist das Array vorhandnen und fehlen in diesem Array dem Gast bereits zugewiesene Gates, werden sie entfernt.**

Beispiel-Request

```
{
  "meta": {
    "failIfExists": false,
    "insertAs": 2
  },
  "guests": [
    {
      "GUEST_FIRST_NAME": "Max",
      "GUEST_LAST_NAME": "Mustermann",
      "GUEST_EMAIL": "max@example.com",
      "GUEST_TICKET_CODE": "ABC123",
      "gates": [1, 3]
    }
  ]
}
```

Response

HTTP Status Codes:

- **422 Formale Fehler**: z.B. fehlendes `meta` Objekt.
- **200 OK**: Alle Datensätze wurden erfolgreich verarbeitet.
- **207 Multi-Status**: Einige Datensätze konnten nicht verarbeitet werden (Details in `results` - s.u.).

Beispiel für HTTP 422

Request

```
{
    "meta": {
        "failIfExists": true,
        "insertAs": 3
```

```
    },
    "guests": [
        {
            "GUEST_TICKET_CODE": "223345",
            "GUEST_FIRST_NAME": "Hans",
            "GUEST_LAST_NAME": "Hermann",
            "gates": [6398]
        },
        {
            "GUEST_TICKET_CODE": "4711",
            "GUEST_FIRST_NAME": "Susi",
            "GUEST_LAST_NAME": "Sorglos",
            "gates": [6399]
        }
    ]
}
```

Response

```
{
    "error": true,
    "message": {
        "meta.insertAs": [
            "The value may not be greater than 2."
        ]
    }
}
```

Erfolgreiche oder Teilerfolgreiche Verarbeitung (HTTP 200, HTTP 207):

Der Response gliedert sich in zwei Bereiche:

1. summary

- `total`: Anzahl der übergebenen Datensätze
- `created`: neu angelegte Gäste
- `updated`: aktualisierte Gäste

- `failed`: fehlerhafte Einträge

2. results

Gibt für jeden Datensatz an, ob die Verarbeitung erfolgreich war oder nicht.

Beispiel für HTTP 207

Request:


```
{
    "meta": {
        "failIfExists": false,
        "insertAs": 2
    },
    "guests": [
        {
            "GUEST_TICKET_CODE": "223344",
            "GUEST_FIRST_NAME": "Hans",
            "GUEST_LAST_NAME": "Hermann",
            "GUEST_EMAIL": "didi@aol.com",
            "gates": [6398]
        },
        {
            "GUEST_FIRST_NAME": "Frank",
            "GUEST_LAST_NAME": "Fuhrmann",
            "GUEST_EMAIL": "ffuhrmann@aol",
            "gates": [6398]
        },
        {
            "GUEST_TICKET_CODE": "4711",
            "GUEST_FIRST_NAME": "Susi",
            "GUEST_LAST_NAME": "Sorglos",
            "gates": [6399,4]
        },
        {
            "TICKET_CODE": "4712",
            "GUEST_FIRST_NAME": "Didi",
            "GUEST_LAST_NAME": "Dröge",
            "gates": [6399]
```

```
        }
    ]
}
```

Response:


```
{
    "summary": {
        "total": 4,
        "created": 0,
        "updated": 1,
        "failed": 3
    },
    "results": [
        {
            "index": 0,
            "ticketCode": "223344",
            "id": 3247987,
            "status": "updated",
            "message": null
        },
        {
            "index": 1,
            "ticketCode": null,
            "id": null,
            "status": "failed",
            "message": {
                "GUEST_EMAIL": [
                    "Please enter a valid email address."
                ]
            }
        },
        {
            "index": 2,
            "ticketCode": "4711",
            "id": 3247986,
            "status": "failed",
            "message": {
                "gates.1": [
                    "The selected gates.1 is invalid."
```

```
                ]
            }
        },
        {
            "index": 3,
            "ticketCode": null,
            "id": null,
            "status": "failed",
            "message": "The following keys are not available: TICKET_CODE"
        }
    ]
}
```