REST API REFERENCE

Conference Submission API

Integration Guide for Third-Party Systems

Version 2.1 | March 2025
Prepared by Avalynn Circe

Overview

The Xerxes Conference Submission API provides programmatic access to the conference management platform, enabling third-party systems to submit abstracts, manage speaker profiles, and retrieve submission status in real time. This guide covers authentication, endpoint reference, request and response formats, error handling, and integration patterns.

> *This API is intended for internal integration teams and approved technology partners. All requests require a valid API key issued by the Xerxes platform administrator.*

Base URL

```
https://api.xerxesconference.com/v2
```

Authentication

All API requests must include a Bearer token in the Authorization header. Tokens are issued per integration and expire after 24 hours. Use the /auth/token endpoint to refresh.

```
Authorization: Bearer <your_api_token>
Content-Type: application/json
X-Client-ID: <your_client_id>
```

> *Never expose API tokens in client-side code or public repositories. Rotate tokens immediately if a breach is suspected. Contact platform-security@xerxesconference.com to revoke a compromised token.*

Endpoints

POST /submissions

Creates a new abstract submission. On success, the platform automatically routes the submission to the appropriate review queue based on the track field value and notifies the assigned program chair via email.

Request Body

Field	Type	Required	Description
track	string	Required	Session track identifier. Must match a value from GET /tracks.
title	string	Required	Abstract title. Maximum 200 characters.
abstract	string	Required	Full abstract text. Maximum 3,000 characters. Plain text only.
authors	array	Required	Array of author objects. See Author Object below.
keywords	array	Optional	Array of keyword strings. Maximum 5 values.
presentation_type	string	Optional	One of: talk, poster, workshop. Defaults to talk.
metadata	object	Optional	Arbitrary key-value pairs for integration-specific data. Not displayed to reviewers.

Author Object

Field	Type	Required	Description
name	string	Required	Full name of the author.
email	string	Required	Contact email. Must be valid RFC 5322 format.
affiliation	string	Optional	Institutional or organizational affiliation.
is_presenter	boolean	Optional	Designates the presenting author. Defaults to false. Exactly one author should be true.

Example Request

```
POST /submissions HTTP/1.1
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/json

{
  "track": "ai-infrastructure",
  "title": "Scaling Retrieval-Augmented Generation in Production",
  "abstract": "This talk presents architectural patterns for deploying...",
  "authors": [
    {
      "name": "Dr. Priya Nair",
      "email": "p.nair@techuniversity.edu",
```

```
      "affiliation": "Tech University",
      "is_presenter": true
    }
  ],
  "keywords": ["RAG", "LLM", "production", "scalability"],
  "presentation_type": "talk"
}
```

Success Response — 201 Created

```
{
  "submission_id": "sub_8f3a2c91d4e7",
  "status": "received",
  "track": "ai-infrastructure",
  "routing": {
    "queue": "technical-review",
    "assigned_chair": "chair-004",
    "notified_at": "2025-03-14T09:22:41Z"
  },
  "created_at": "2025-03-14T09:22:41Z"
}
```

GET /submissions/{submission_id}

Returns the current status and full details of an existing submission. Use this endpoint to poll for status changes after an initial POST.

Path Parameters

Field	Type	Required	Description
submission_id	string	Required	The submission_id returned by POST /submissions.

Example Response — 200 OK

```
{
  "submission_id": "sub_8f3a2c91d4e7",
  "status": "under_review",
  "track": "ai-infrastructure",
  "title": "Scaling Retrieval-Augmented Generation in Production",
  "authors": [...],
  "review_stage": 2,
  "created_at": "2025-03-14T09:22:41Z",
  "updated_at": "2025-03-16T14:05:18Z"
}
```

Error Handling

All errors follow a consistent structure. The code field maps to the HTTP status code. The message field provides a human-readable explanation. The details field, when present, contains field-level validation errors.

Error Response Structure

```
{
  "error": {
    "code": 422,
    "type": "validation_error",
    "message": "Request body contains invalid fields.",
    "details": [
      { "field": "authors[0].email", "issue": "Invalid email format." },
      { "field": "track", "issue": "Unknown track identifier: 'ai-infra'." }
    ]
  }
}
```

HTTP Status Code Reference

Code	Status	Description
`200`	OK	Request succeeded. Response body contains requested resource.
`201`	Created	Resource created successfully. Response body contains the new resource.
`400`	Bad Request	Malformed request syntax. Check JSON structure and Content-Type header.
`401`	Unauthorized	Missing or invalid Authorization token. Re-authenticate and retry.
`403`	Forbidden	Valid token but insufficient permissions for this operation.
`404`	Not Found	The requested resource does not exist.
`422`	Unprocessable Entity	Request is well-formed but contains semantic errors. See details array.
`429`	Too Many Requests	Rate limit exceeded. Retry after the interval in Retry-After header.
`500`	Internal Server Error	Unexpected server-side error. Contact platform support if persistent.

Rate Limits

The API enforces rate limits per client ID. Limits reset on a rolling 60-second window.

Field	Type	Required	Description

`POST /submissions`	—	—	100 requests per minute per client ID.
`GET /submissions/*`	—	—	500 requests per minute per client ID.
`All other endpoints`	—	—	200 requests per minute per client ID.

> *When a 429 is returned, check the Retry-After response header for the number of seconds to wait before retrying. Implementing exponential backoff is strongly recommended for production integrations.*

Webhooks

Register a webhook endpoint to receive real-time push notifications when submission status changes, rather than polling the GET /submissions endpoint.

Registering a Webhook

```
POST /webhooks

{
  "url": "https://your-system.example.com/hooks/xerxes",
  "events": ["submission.status_changed", "submission.decision_released"],
  "secret": "your_hmac_signing_secret"
}
```

Verifying Webhook Signatures

Every webhook POST includes an X-Xerxes-Signature header. Compute an HMAC-SHA256 of the raw request body using your registered secret and compare it to the header value to verify authenticity.

```
// Node.js example
const crypto = require('crypto');

function verifySignature(rawBody, secret, signatureHeader) {
  const computed = crypto
    .createHmac('sha256', secret)
    .update(rawBody)
    .digest('hex');
  return crypto.timingSafeEqual(
    Buffer.from(computed),
    Buffer.from(signatureHeader)
  );
}
```

Changelog

Field	Type	Required	Description
v2.1	March 2025	—	Added metadata field to POST /submissions. Added webhook signature verification section.
v2.0	October 2024	—	Migrated authentication to Bearer tokens. Deprecated API key query parameter (removed in v2.1).
v1.4	June 2024	—	Added presentation_type field. Added GET /tracks endpoint.