

Conecta tu aplicación con Feending

Gestiona usuarios, sincroniza cambios, publica contenido y recibe webhooks desde tus aplicaciones.

La API de Feending utiliza la ruta base /ext/v2. Los siguientes apartados detallan la autenticación, los permisos, la paginación y el control de versiones, con ejemplos de peticiones y respuestas.

Por dónde empezar

Elemento	Cómo se utiliza
Conectar	Comprueba la clave, los permisos y los límites de tu plataforma.
Gestionar personas	Busca usuarios, crea cuentas y actualiza o desactiva perfiles.
Mantener los datos al día	Lee los cambios con cursor, incluidas bajas. Combina esta consulta con los webhooks.
Dar acceso	Solicita un enlace temporal para una persona autenticada en tu portal.
Conectar actividad	Gestiona eventos, inscripciones y publicaciones.
Recibir avisos	Configura un webhook y consulta el resultado de sus entregas.

Los ejemplos utilizan datos ficticios. Sustituye los valores entre dobles llaves por los de tu conexión.

Guía por bloques

Conexión y contrato de la API

Autenticación, permisos, respuestas, escrituras seguras y tratamiento de errores.

Usuarios, grupos y acceso

Crear y mantener cuentas, gestionar pertenencias y dar acceso desde un portal autenticado.

Sincronización de datos

Desde la primera carga hasta la recuperación de cambios: paginación, cursores, bajas e interrupciones.

Actividades, inscripciones y contenido

Consultar y gestionar eventos, participantes y publicaciones con sus estados y reglas de negocio.

Webhooks y catálogo de eventos

Configurar receptores, verificar firmas, recuperar entregas y consultar los eventos y sus datos.

Validación de la integración

Comprobar el flujo completo y los casos de error antes de dar la conexión por terminada.

Conexión y contrato de la API

Autenticación, permisos, respuestas, escrituras seguras y tratamiento de errores.

En este bloque

Dónde se adapta el formato de cada sistema

Ejemplo de transformación en ambos sentidos

Errores, reintentos y cambios de formato

Primera conexión y permisos

Respuestas y paginación

Reintentos y cambios simultáneos

Errores y recuperación

Dónde se adapta el formato de cada sistema

La API de Feending recibe el esquema documentado de cada operación. El componente que la consume debe transformar el contrato de origen y ejecutar la operación correspondiente.

Modalidad	Dónde se ejecuta el adaptador	Qué mantiene su responsable
API directa	Backend del cliente	Mapeo, credenciales, llamadas y recuperación
Servicio intermedio	Plataforma de automatización o servicio de integración	Flujos, conexiones, transformaciones y seguimiento de entregas
Conector de Feending	Servicio definido para la integración	Contrato de entrada, adaptador del proveedor y operación del conector según el alcance acordado

Ejemplos de herramientas para el servicio intermedio: Microsoft Power Automate, con un conector personalizado; Make, con su aplicación HTTP; Zapier, con sus herramientas de peticiones API; y n8n, con el nodo HTTP Request. Configura método, URL, cabeceras y cuerpo según el contrato de Feending. Estas opciones no presuponen un conector nativo de Feending.

Contrato de entrada del adaptador

Si el sistema de origen emite su propio webhook, lo envía al receptor del adaptador. Ese receptor comprueba la autenticidad con el mecanismo del proveedor, valida el mensaje y lo convierte antes de llamar a Feending. La URL y la autenticación de este receptor pertenecen a la integración concreta; no existe una URL universal que interprete cualquier cuerpo del cliente.

Cuando el origen no emite avisos, el adaptador consulta su API con la periodicidad y el mecanismo incremental que ese sistema permita.

Contrato de salida y correspondencia de registros

Conserva la relación entre la clave del origen y el identificador de Feending. Utiliza `externalReferences` y la búsqueda por `externalSystem` y `externalId` para resolverla. El nombre de sistema externo debe estar autorizado en la conexión. Un resultado vacío requiere decidir si corresponde crear una cuenta; no demuestra por sí solo que haya que borrar nada.

Configura por campo su origen autorizado, tipo, conversión, obligatoriedad y tratamiento de valores ausentes o nulos. En PATCH, omitir un campo conserva su valor; enviar null solo lo borra si el contrato de ese campo lo admite. La API rechaza los campos que no pertenecen a la operación.

Ejemplo de transformación en ambos sentidos

Ejemplo ilustrativo: el sistema de personal publica este cambio. Los nombres de campos pertenecen a ese sistema, no al contrato de Feending.

```
{
  "change_id": "CHANGE-458",
  "employee_id": "EMP-123",
  "given_name": "Ana",
  "surname": "García",
  "team": "Ventas",
  "enabled": true
}
```

Entrada: sistema de personal → Feending

1. Valida el aviso y resuelve EMP-123 con GET /ext/v2/users?externalSystem=rrhh&externalId=EMP-123. rrhh representa aquí el espacio externo autorizado.
2. Lee el usuario por su identificador para obtener el ETag actual. Traduce los campos y envía este cuerpo a PATCH /ext/v2/users/{identifier}, con api-key, If-Match e Idempotency-Key.

```
{
  "firstName": "Ana",
  "lastName": "García",
  "department": "Ventas",
  "status": "active"
}
```

Conserva la clave de idempotencia al repetir la misma solicitud. Ante un 412, relee el usuario y resuelve el conflicto antes de preparar una nueva operación. El ejemplo presupone que el sistema de personal es responsable de esos campos y que enabled determina el acceso a la comunidad.

Salida: Feending → sistema del cliente

Recibe un webhook documentado para el cambio o recupéralo desde el registro de cambios. Resuelve el identificador externo y convierte únicamente los campos cuyo origen autorizado sea Feending. Por ejemplo, si el cliente guarda el teléfono en mobile, la adaptación de phone puede producir:

```
{"employee_id":"EMP-123","mobile":"+34 600 000 000"}
```

Envía el resultado a la operación de actualización documentada por el sistema del cliente, con sus propias credenciales. Su URL, método y respuesta dependen de ese sistema.

Errores, reintentos y cambios de formato

Evitar duplicados y bucles

Registra la identidad del cambio de origen y la operación de destino. Deduplica los avisos repetidos; reutiliza la clave de idempotencia para reintentar una misma solicitud a Feending. No envíes de vuelta un campo que acaba de actualizar su sistema responsable. Si ambos sistemas permiten editarlo, configura una regla de conflicto explícita.

Confirmar la recepción y completar la operación

Guarda de forma duradera un aviso válido antes de confirmar su recepción. La aceptación del webhook y la actualización del sistema de destino son estados distintos. Conserva el resultado de la operación, sus intentos y el error pendiente; utiliza el registro de cambios para recuperar periodos sin conexión.

Validar antes de escribir

Un identificador desconocido, una fecha ambigua o un estado sin correspondencia requieren tratamiento explícito. No conviertas valores desconocidos en una baja ni completes datos obligatorios con valores inventados. Aparta el mensaje con un motivo legible y permite volver a procesarlo una vez corregida la causa.

Proteger y mantener la conexión

Guarda las credenciales en el servicio que ejecuta el adaptador, fuera del navegador y de los registros. Separa las conexiones y correspondencias por comunidad. Mantén versionadas las reglas de transformación y verifica sus efectos cuando cambie la API de cualquiera de los sistemas.

Prueba de aceptación	Resultado esperado
Mismo aviso recibido dos veces	Una sola modificación efectiva
Campo omitido frente a campo nulo	Conservación o borrado según el contrato
Registro externo desconocido	Alta o error conforme a la regla definida
Actualizaciones simultáneas	Conflicto detectado y resuelto sin sobrescribir a ciegas
Destino temporalmente caído	Reintento recuperable y estado observable
Cambio procedente del otro sistema	No genera un intercambio indefinido de actualizaciones

Primera conexión y permisos

```
GET {{FEENDING_BASE_URL}}/ext/v2/connection
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

Respuesta 200

```
{
  "platform":{"identifier":"PLATFORM-ejemplo"},
  "permissions":["users:read","users:write","users:link","changes:read"],
  "limits":{"
    "defaultPageSize":50,"maxPageSize":200,
    "idempotencyRetentionSeconds":86400,
    "changesRetentionSeconds":2592000
  }}
}
```

Los límites de esta respuesta son ilustrativos. Utiliza los valores devueltos por tu conexión; no presupongas que son iguales en todas las plataformas.

Tu servidor envía la clave en cada petición. La clave identifica la plataforma y los permisos concedidos. Guárdala en el gestor de secretos y no la incluyas en el código público de una web o aplicación móvil.

Elemento	Cómo se utiliza
Datos	users:read/write, groups:read/write, events:read/write, registrations:read/write y posts:read/write. Cada operación exige read o write según corresponda.
Enlace de cuentas	users:link para POST /users/links. El sistema de origen debe estar autorizado en la conexión. Consultar o modificar el perfil requiere sus propios permisos users:read o users:write.
Integración	changes:read, sso:issue, webhooks:read/write. Emitir un enlace de usuario requiere sso:issue.
Errores	401 para una clave ausente, inválida o revocada. 403 para una clave válida sin el permiso necesario.
Rotación	Sustituye la clave en tus servicios y comprueba GET /connection. No presupongas que la anterior seguirá funcionando.

Respuestas y paginación

Envía JSON con nombres camelCase y fechas ISO 8601 en UTC. Conserva los identificadores recibidos y codifícalos cuando formen parte de una URL.

Elemento	Cómo se utiliza
Consulta individual	200 con {"data":{...}} y ETag. Un recurso inexistente o fuera de tu plataforma devuelve 404.
Creación	201 con data, Location y ETag del nuevo recurso.
Actualización	200 con data y nuevo ETag. PATCH cambia solo los campos enviados. null borra únicamente campos que admiten ese valor.
Listado	200 con data como array, nextCursor y hasMore. Utiliza limit dentro del máximo de GET /connection.
Campos nuevos	Acepta propiedades adicionales en respuestas. En escrituras, un campo desconocido devuelve validation_error para detectar errores del emisor.

```
GET {{FEENDING_BASE_URL}}/ext/v2/users?limit={{PAGE_SIZE}}&cursor={{CURSOR_URL_ENCODED}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

```
{
  "data": [{
    "identifier": "USER-A", "email": "ana@example.test",
    "status": "active", "version": 3
  }],
  "nextCursor": "<cursor-opaco>", "hasMore": true
}
```

Repite la consulta con nextCursor y los mismos filtros. Al terminar, hasMore=false y nextCursor=null. No construyas el cursor ni lo uses como cursor del registro de cambios: son mecanismos distintos.

Reintentos y cambios simultáneos

Una clave por operación

En las escrituras envía Idempotency-Key con un identificador generado por tu aplicación. Si pierdes la respuesta, repite la misma petición con la misma clave y el mismo cuerpo. Dentro del periodo de conservación, Feending devuelve el resultado guardado sin repetir efectos.

La misma clave con otro método, ruta o cuerpo produce 409 idempotency_conflict. Si la primera petición sigue en curso, devuelve 409 request_in_progress con Retry-After. Una nueva acción necesita otra clave.

Editar la versión que has consultado

Copia el ETag de la consulta individual en If-Match, incluidas las comillas. Si otro proceso modificó el recurso, Feending devuelve 412. Si falta la condición, devuelve 428.

```
PATCH {{FEENDING_BASE_URL}}/ext/v2/users/{{USER_ID}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
If-Match: "3"
```

```
{"department": "Operaciones"}
```

```
HTTP/1.1 200 OK
ETag: "4"
Content-Type: application/json
```

```
{"data":{"identifier":"USER-A","department":"Operaciones",
        "status":"active","version":4}}
```

Ante un 412, consulta otra vez, decide qué campos mantener y envía la edición con el nuevo ETag y una nueva clave de operación. No sobrescribas automáticamente cambios ajenos.

Errores y recuperación

HTTP/1.1 412 Precondition Failed
Content-Type: application/json

```
{"error":{"code":"version_conflict",  
  "message":"El recurso ha cambiado desde la última consulta.",  
  "requestId":"REQUEST-ejemplo","details":{"currentVersion":4}}}
```

Elemento	Cómo se utiliza
400 invalid_cursor / invalid_filter	Corrige los parámetros o reinicia la consulta con los filtros apropiados.
401 invalid_credentials	Comprueba la clave y el origen de la API.
403 insufficient_scope	La conexión no tiene el permiso necesario.
404 resource_not_found	El recurso no existe o no está disponible en la plataforma.
409 conflicto	Utiliza error.code para distinguir duplicados, una operación incompatible o una clave idempotente reutilizada.
410 cursor_expired	Crea un checkpoint y repite la carga completa.
412 version_conflict / 428 precondition_required	Consulta la versión actual o añade If-Match.
422 validation_error	details.fields indica campos y causas. Corrige el cuerpo.
429 rate_limited	Respetar Retry-After y reducir la frecuencia.
5xx / red	Reintenta con espera gradual. Conserva Idempotency-Key para escrituras: el cambio puede haberse aplicado.

Utiliza code para programar el tratamiento y requestId para localizar la petición. No dependas del texto traducible de message.

Usuarios, grupos y acceso

Crear y mantener cuentas, gestionar pertenencias y dar acceso desde un portal autenticado.

En este bloque

Crear una cuenta y guardar su enlace

Buscar y consultar una persona

Enlazar cuentas que ya existen

Reutilizar el enlace y resolver conflictos

Crear una cuenta desde tu sistema

Actualizar, desactivar y reactivar

Consultar grupos y gestionar sus miembros

Dar acceso desde tu portal

Crear una cuenta y guardar su enlace

La clave de correspondencia es la comunidad, el sistema de origen y su identificador externo. El identificador de Feending tiene el formato USER-{GUID}; guarda siempre el valor devuelto por la API.

Crear al usuario con su referencia externa

Ejemplo: rrhh es el nombre autorizado para el sistema de personal y EMP-123 es su identificador estable del empleado. Incluye la referencia en el alta:

```
POST {{FEENDING_BASE_URL}}/ext/v2/users
api-key: {{FEENDING_API_KEY}}
Idempotency-Key: {{OPERATION_ID}}
Content-Type: application/json
```

```
{
  "email": "ana@example.test",
  "firstName": "Ana",
  "lastName": "García",
  "externalReferences": [
    {"system": "rrhh", "id": "EMP-123"}
  ]
}
```

La respuesta es 201 Created. Esta proyección muestra los campos que establecen el enlace; la respuesta incluye también los demás campos del usuario:

```
{
  "data": {
    "identifier": "USER-550e8400-e29b-41d4-a716-446655440000",
    "externalReferences": [
      {"system": "rrhh", "id": "EMP-123"}
    ]
  }
}
```

Dónde queda guardado

Feending conserva externalReferences asociadas a esa cuenta dentro de la comunidad. La integración puede conservar también el identificador devuelto para evitar una búsqueda en cada actualización. Guarda la comunidad y el sistema junto al identificador externo; EMP-123 por sí solo no identifica una persona entre todos los clientes.

Una misma pareja system/id no puede asignarse a dos usuarios de la misma comunidad. Un conflicto de referencia devuelve 409: investiga la asociación existente antes de intentar crear otra cuenta. El alta y sus referencias se confirman en una sola operación.

Buscar y consultar una persona

Para localizar la cuenta de EMP-123 utiliza la referencia guardada en Feending. Después opera con el identificador de Feending, que tiene el formato USER-{GUID}.

1. Buscar por la referencia guardada en Feending

```
GET {{FEENDING_BASE_URL}}/ext/v2/users?externalSystem=rrhh&externalId=EMP-123
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

La búsqueda devuelve un listado. Esta proyección omite otros campos del usuario para destacar la correspondencia:

```
{
  "data": [{
    "identifier": "USER-550e8400-e29b-41d4-a716-446655440000",
    "externalReferences": [{"system": "rrhh", "id": "EMP-123"}]
  }],
  "hasMore": false,
  "nextCursor": null
}
```

2. Consultar y modificar la cuenta localizada

Guarda data[0].identifier como FEENDING_USER_ID y consulta el recurso individual:

```
GET {{FEENDING_BASE_URL}}/ext/v2/users/{{FEENDING_USER_ID}}
api-key: {{FEENDING_API_KEY}}
```

Esta respuesta contiene data como objeto y el ETag en la cabecera HTTP. Usa ese ETag en If-Match al enviar el PATCH a la misma URL, junto con una clave Idempotency-Key. No busques por el nombre de la persona ni reconstruyas el GUID a partir del identificador externo.

3. Resolver un cambio en sentido inverso

Si recibes el identificador de Feending desde un evento o desde el registro de cambios, consulta las externalReferences del recurso o del usuario. Selecciona la referencia cuyo system corresponda al destino y utiliza su id, por ejemplo EMP-123. Una persona puede tener referencias de varios sistemas; no uses sin comprobar la primera de la lista.

data: [] significa que no hay una cuenta accesible con esa referencia. No demuestra que la persona no exista; podría tener una cuenta todavía sin enlazar. El listado también admite email exacto normalizado, status, groupIdentifier y limit/cursor. Una búsqueda por correo no autentica a la persona.

Enlazar cuentas que ya existen

Envía el identificador de vuestro sistema y un dato común para localizar la cuenta. Feending guarda la correspondencia en su base de datos; las sincronizaciones siguientes utilizan los identificadores.

Emparejamiento inicial por email o DNI/NIF/NIE

Esta operación requiere `users:link` y que `system` esté autorizado para tu conexión. La comunidad se obtiene de la credencial. Utiliza `match.email`, `match.nif` o ambos. El campo `nif` admite el DNI/NIF/NIE que ya figure en la cuenta de Feending.

```
POST {{FEENDING_BASE_URL}}/ext/v2/users/links
api-key: {{FEENDING_API_KEY}}
Idempotency-Key: {{OPERATION_ID}}
Content-Type: application/json
```

```
{
  "system": "rrhh",
  "id": "EMP-123",
  "match": {"email": "ana@example.test", "nif": "00123456Z"}
}
```

Si envías ambos datos, deben coincidir en la misma cuenta. El correo se compara sin distinguir mayúsculas ni espacios en los extremos. En el documento se eliminan espacios y separadores, se convierten las letras a mayúsculas y se conservan los ceros iniciales. La comparación es exacta después de normalizar.

Respuesta 200: enlace guardado

```
{
  "data": {
    "userIdentifier": "USER-550e8400-e29b-41d4-a716-446655440000",
    "externalReference": {"system": "rrhh", "id": "EMP-123"}
  }
}
```

Guarda `data.userIdentifier` para consultar o actualizar a esa persona. También puedes recuperar el enlace con `GET /ext/v2/users?externalSystem=rrhh&externalId=EMP-123`. La referencia aparece en `externalReferences` del usuario, lo que permite resolver la correspondencia en sentido inverso.

El enlace no cambia el perfil, el estado ni los permisos de la cuenta. Puede asociar una cuenta desactivada, pero no reactiva su acceso. El DNI/NIF/NIE no se devuelve en la respuesta; comparar este dato no autentica a la persona.

Reutilizar el enlace y resolver conflictos

La pareja system/id identifica una única cuenta dentro de la comunidad. Feending consulta primero esa relación y solo busca por email o documento si todavía no existe.

Cuando la relación ya está guardada

Puedes resolverla con la misma operación, enviando únicamente la referencia:

```
POST {{FEENDING_BASE_URL}}/ext/v2/users/links
api-key: {{FEENDING_API_KEY}}
Idempotency-Key: {{NEW_OPERATION_ID}}
Content-Type: application/json
```

```
{"system": "rrhh", "id": "EMP-123"}
```

Devuelve el mismo userIdentifier con estado 200. No crea otra asociación ni vuelve a emparejar por email o documento. Si todavía incluyes criterios válidos en match, prevalece el enlace existente. Un cambio de correo o DNI/NIF/NIE nunca reasigna automáticamente EMP-123 a otra persona.

Elegir una cuenta después de una revisión

Si ya has confirmado qué cuenta corresponde, sustituye match por userIdentifier:

```
{
  "system": "rrhh", "id": "EMP-123",
  "userIdentifier": "USER-550e8400-e29b-41d4-a716-446655440000"
}
```

La cuenta debe pertenecer a vuestra comunidad. No combines ambos selectores. Si la referencia ya apunta a otra cuenta, esta petición falla: no sobrescribe el enlace.

Errores y siguiente acción

HTTP / código	Qué debe hacer la integración
404 · user_match_not_found	No hay una cuenta que cumpla todos los criterios. Revisa los datos; crea una cuenta solo si corresponde a tu proceso de alta.
409 · user_match_ambiguous	Hay varias coincidencias. Detén ese registro y confirma el USER-{GUID} correcto antes de enlazar.
409 · external_reference_conflict o linked_user_unavailable	La referencia apunta a otra cuenta o su cuenta fue eliminada. Revisa la relación; no reutilices el identificador para otra persona.
422 · match_required	La referencia es nueva. Incluye match o un userIdentifier confirmado.

Ante 409 request_in_progress, respeta Retry-After y repite la petición con la misma clave. Conserva la clave y el cuerpo original después de un timeout. Una nueva petición para la misma referencia también devuelve el enlace sin duplicarlo. No se requiere If-Match: esta operación añade una relación y no permite cambiarla ni eliminarla.

Crear una cuenta desde tu sistema

POST {{FEENDING_BASE_URL}}/ext/v2/users

api-key: {{FEENDING_API_KEY}}

Accept: application/json

Content-Type: application/json

Idempotency-Key: {{OPERATION_ID}}

```
{
  "email": "ana@example.test", "firstName": "Ana", "lastName": "Ejemplo",
  "department": "Operaciones",
  "externalReferences": [{"system": "rrhh", "id": "EMP-1042"}]
}
```

Elemento	Cómo se utiliza
Obligatorios	email, firstName y lastName. El servidor normaliza y valida el correo.
Opcionales	department, jobTitle y phone, anulables; externalReferences, lista de pares system/id únicos.
Estado	El alta crea una cuenta active, sujeta a la aprobación y activación que exija la plataforma.
Efectos	No inicia sesión, no inscribe en actividades ni envía invitaciones implícitas. Genera user-created.
Protegidos	No admite contraseñas, tokens, roles administrativos ni fechas de aprobación en este cuerpo.

HTTP/1.1 201 Created

Location: /ext/v2/users/USER-A

ETag: "1"

Content-Type: application/json

```
{"data":{"identifier":"USER-A","email":"ana@example.test",
  "firstName":"Ana","lastName":"Ejemplo",
  "status":"active","version":1}}
```

Guarda identifier. Un correo o referencia externa ya asociado a otra cuenta devuelve 409 email_conflict o external_reference_conflict. Comprueba la vinculación antes de modificar una cuenta existente.

Actualizar, desactivar y reactivar

```
PATCH {{FEENDING_BASE_URL}}/ext/v2/users/{{USER_ID}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
If-Match: {{RESOURCE_ETAG}}
```

```
{"department":"Finanzas","jobTitle":"Responsable de área"}
```

El PATCH admite firstName, lastName, department, jobTitle y phone. Los tres últimos permiten null. Email, referencias externas y permisos no se modifican con este cuerpo. Devuelve el usuario con un ETag nuevo y genera user-updated.

Desactivar

```
PATCH {{FEENDING_BASE_URL}}/ext/v2/users/{{USER_ID}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
If-Match: {{RESOURCE_ETAG}}
```

```
{"status":"inactive"}
```

Impide nuevos accesos y revoca las sesiones de Feending de la persona. Conserva identificador e historial. Genera user-deactivated y una entrada en el registro de cambios.

Reactivar

```
PATCH {{FEENDING_BASE_URL}}/ext/v2/users/{{USER_ID}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
If-Match: {{RESOURCE_ETAG}}
```

```
{"status":"active"}
```

Restablece la posibilidad de acceder si se cumplen las demás condiciones de la cuenta. No recupera sesiones anteriores ni aprueba automáticamente al usuario. Genera user-reactivated.

La baja operativa no elimina definitivamente los datos. Para una eliminación permanente se utiliza el procedimiento de eliminación aplicable.

Consultar grupos y gestionar sus miembros

```
GET {{FEENDING_BASE_URL}}/ext/v2/groups?limit={{PAGE_SIZE}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

```
GET {{FEENDING_BASE_URL}}/ext/v2/groups/{{GROUP_ID}}/members?limit={{PAGE_SIZE}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

Los grupos incluyen identifier, name y version. El listado de miembros devuelve sus identificadores. Ambos listados son paginados.

Añadir un miembro

```
PUT {{FEENDING_BASE_URL}}/ext/v2/groups/{{GROUP_ID}}/members/{{USER_ID}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
```

```
{}
```

```
{"data":{"groupIdentifier":"GROUP-A","userIdentifier":"USER-A",
      "present":true,"version":2}}
```

Retirar un miembro

```
DELETE {{FEENDING_BASE_URL}}/ext/v2/groups/{{GROUP_ID}}/members/{{USER_ID}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Idempotency-Key: {{OPERATION_ID}}
```

Responde 204 cuando la pertenencia ya no existe. Añadir a alguien ya presente o retirar a alguien ausente no repite efectos. Estas operaciones no envían notificaciones implícitas.

Se requiere groups:write y autorización sobre el grupo. No se pueden conceder roles privilegiados por esta vía. Un cambio efectivo genera group-member-added o group-member-removed y aparece en el registro groupMemberships.

Dar acceso desde tu portal

Tu aplicación autentica a la persona y comprueba que puede entrar en la plataforma. Tu backend obtiene su identifier de la relación de cuentas guardada en el servidor y solicita un enlace:

```
POST {{FEENDING_BASE_URL}}/ext/v2/auth/login-links
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
```

```
{"userIdentifier":"USER-A"}
```

```
HTTP/1.1 201 Created
Content-Type: application/json
```

```
{"data":{"loginUrl":"<enlace-completo>",
        "expiresAt":"<fecha-UTC-de-caducidad>"}}
```

Redirige el navegador a data.loginUrl. El token dura cinco minutos. Repetir la petición con la misma Idempotency-Key devuelve el mismo enlace sin ampliar su vigencia; un nuevo acceso utiliza otra clave.

Elemento	Cómo se utiliza
Permiso	sso:issue. El navegador no puede elegir libremente el identificador de otra persona.
Destino opcional	returnUrl: ruta interna que comienza por /. No se aceptan URLs absolutas ni rutas que empiecen por //.
Cuenta desactivada	403 user_inactive. Para condiciones de activación o aprobación pendientes: 403 user_not_ready.
Baja	Desactivar la cuenta revoca las sesiones. Reactivarla exige un nuevo acceso.

Permite aprovechar un portal conectado a Entra ID, Active Directory u otro proveedor. La federación directa OIDC/SAML desde la pantalla de Feending se trata por separado: la ampliación de endpoints no la incorpora automáticamente.

Sincronización de datos

Desde la primera carga hasta la recuperación de cambios: paginación, cursores, bajas e interrupciones.

En este bloque

Dos cursores con funciones distintas

Ejemplo: recorrer un listado

Sincronizar cambios, incluidas bajas

Guardar el progreso de la sincronización

Ejemplo: primera carga y siguientes lecturas

Interrupciones, bajas y webhooks

Dos cursores con funciones distintas

Un cursor es un valor opaco que devuelve Feending para continuar una lectura. Tu aplicación lo guarda y lo devuelve sin interpretarlo, recortarlo ni construirlo a partir de una fecha. Codifícalo como parámetro de URL con la biblioteca HTTP de tu lenguaje.

Elemento	Cómo se utiliza
Cursor de listado	Recorre un conjunto de usuarios, grupos u otros recursos por páginas. Sirve para la carga inicial o una consulta. Al finalizar, nextCursor es null.
Cursor de cambios	Marca hasta dónde has leído el registro de modificaciones. Sirve para continuar la sincronización en la próxima ejecución. Se conserva incluso cuando hasMore es false.
No son intercambiables	El cursor recibido de GET /users se devuelve a GET /users. El recibido de /changes/checkpoints o GET /changes se utiliza en GET /changes.
Ámbito	Mantén un estado independiente por plataforma, recurso y conjunto de filtros. No compartas un cursor entre conexiones ni cambies los filtros a mitad del recorrido.

Qué significa hasMore

true indica que queda otro bloque disponible y debes continuar. false indica que has alcanzado el final disponible de esa lectura. En el registro de cambios no significa que la sincronización haya terminado para siempre: más adelante pueden aparecer nuevos cambios.

Qué debes guardar

Para una sincronización continua conserva el cursor de cambios confirmado, los identificadores y versiones locales, y el estado de la carga inicial. El cursor de un listado es temporal y nunca sustituye al cursor de cambios.

Ejemplo: recorrer un listado

Esta consulta ilustra un listado de tres usuarios en dos páginas. Los valores PAGINA_2 y CAMBIOS_0 de esta guía son etiquetas explicativas; en una integración real se utiliza el valor exacto que devuelve el servidor.

1. Primera página, sin cursor

```
GET {{FEENDING_BASE_URL}}/ext/v2/users?limit=2
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

```
{"data":[{"identifier":"USER-A","version":3},
        {"identifier":"USER-B","version":6}],
  "nextCursor":"PAGINA_2","hasMore":true}
```

2. Guarda los usuarios y solicita la siguiente página

```
GET {{FEENDING_BASE_URL}}/ext/v2/users?limit=2&cursor=PAGINA_2
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

```
{"data":[{"identifier":"USER-C","version":1}],
  "nextCursor":null,"hasMore":false}
```

Los objetos están abreviados para mostrar la paginación. Guarda los registros de cada página antes de continuar. El tamaño de página de este ejemplo debe respetar los límites que devuelve GET /connection.

Con hasMore=false termina el listado. Para mantener esa copia actualizada necesitas el registro de cambios descrito a continuación. No uses el último usuario ni su fecha como sustitutos del cursor.

Una carga inicial puede coincidir con ediciones de usuarios. Por eso primero se crea un punto de partida del registro de cambios y, después del listado, se recuperan las modificaciones posteriores a ese punto.

Sincronizar cambios, incluidas bajas

1. Guarda un punto de partida

```
POST {{FEENDING_BASE_URL}}/ext/v2/changes/checkpoints
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
```

```
{}
```

```
HTTP/1.1 201 Created
Content-Type: application/json
```

```
{"data":{"cursor":"<punto-inicial>",
        "expiresAt":"<fecha-UTC-de-caducidad>"}}
```

Después recorre el listado paginado de usuarios. Al terminar, consume cambios desde el punto inicial para recuperar modificaciones que hayan ocurrido durante la carga.

2. Lee los cambios

```
GET {{FEENDING_BASE_URL}}/ext/v2/changes?resource=users&cursor={{CURSOR_URL_ENCODED}}&limit={{PAGE_SIZE}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

```
{
  "data":[
    {
      "id":"CHANGE-1",
      "resource":"users",
      "operation":"update",
      "identifier":"USER-A",
      "version":4,
      "occurredAt":"2026-10-02T10:10:00Z",
      "data":{"identifier":"USER-A","status":"inactive","version":4}},
    {
      "id":"CHANGE-2",
      "resource":"users",
      "operation":"delete",
      "identifier":"USER-B",
      "version":7,
      "occurredAt":"2026-10-02T10:11:00Z",
      "data":null}
  ],
  "nextCursor":"<siguiente-punto>",
  "hasMore":false
}
```

El ejemplo reduce los campos del primer usuario. update contiene el estado permitido para tu conexión. delete conserva identifier y version aunque el recurso ya no pueda consultarse.

Guardar el progreso de la sincronización

1. **Aplica por identifier y version.** Un mensaje repetido o antiguo no debe sobrescribir una versión más reciente.
2. **Guarda los cambios y el cursor juntos.** Confirma ambos en la misma transacción cuando sea posible. En otro caso, garantiza que repetir el lote no duplique efectos.
3. **Continúa mientras hasMore sea true.** Repite con nextCursor y el mismo filtro. Cuando sea false, conserva también nextCursor para la siguiente consulta.
4. **Recupera una interrupción desde el último cursor confirmado.** No avances hasta haber guardado correctamente el lote.

Cursor caducado

Un cursor fuera del periodo de conservación devuelve 410 cursor_expired. Crea un checkpoint nuevo y repite la carga completa. No lo sustituyas por una fecha local.

Recursos admitidos

resource admite users, groups, groupMemberships, events, registrations y posts. Necesitas changes:read y permiso de lectura sobre el recurso. Cambiar plataforma, filtros o permisos exige una nueva sincronización.

Orden

El registro conserva el orden de incorporación de los cambios. La versión del recurso determina si un cambio es más reciente; no lo deduzcas solo de occurredAt. Los cambios de la API y de la aplicación comparten esta semántica.

Webhooks y consultas

El webhook permite reaccionar pronto; el cursor permite recuperar cambios no procesados. Ambos deben actualizar la misma copia local por identifier y version. Una comparación periódica comprueba que los sistemas siguen coincidiendo.

Ejemplo: primera carga y siguientes lecturas

Quieres mantener una copia de los usuarios de Feending en tu aplicación. El orden de las operaciones evita perder los cambios que se produzcan mientras descargas el listado.

1. **Crea el punto de partida.** POST /changes/checkpoints devuelve CAMBIOS_0. Guárdalo como punto inicial y marca la carga como incompleta.
2. **Descarga los usuarios.** Recorre GET /users con sus propios cursores hasta hasMore=false. Aplica cada usuario por identifier y version.
3. **Recupera lo ocurrido durante la carga.** Consulta GET /changes con resource=users y CAMBIOS_0.

```
GET {{FEENDING_BASE_URL}}/ext/v2/changes?resource=users&cursor=CAMBIOS_0&limit={{PAGE_SIZE}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

```
{
  "data": [
    {
      "id": "CHANGE-1",
      "resource": "users",
      "operation": "update",
      "identifier": "USER-A",
      "version": 4,
      "occurredAt": "2026-10-02T10:10:00Z",
      "data": {
        "identifier": "USER-A",
        "status": "inactive",
        "version": 4
      }
    }
  ],
  "nextCursor": "CAMBIOS_1",
  "hasMore": false
}
```

El objeto de usuario está abreviado. Si ya descargaste USER-A en versión 4, esta entrada no debe volver a ejecutar acciones. Si conservas la versión 3, aplica la versión 4.

4. **Confirma el progreso.** Guarda el resultado del lote y CAMBIOS_1 juntos. Al alcanzar hasMore=false puedes dar por completada la carga inicial.
5. **Consulta de nuevo más adelante.** Usa CAMBIOS_1, sin repetir el listado completo.

```
{
  "data": [],
  "nextCursor": "CAMBIOS_2",
  "hasMore": false
}
```

Una respuesta vacía es válida: no hay cambios pendientes para esa lectura. Conserva el nextCursor devuelto y utilízalo en la siguiente consulta. No esperes a recibir datos para guardar el progreso.

Interrupciones, bajas y webhooks

Si el proceso se interrumpe

Supón que tienes confirmado CAMBIOS_1. Lees un lote que devuelve CAMBIOS_2, pero el proceso falla antes de guardar todo. Reinicia desde CAMBIOS_1. Algunos cambios pueden repetirse: aplica solo versiones posteriores a la que conservas y confirma CAMBIOS_2 después de completar el lote.

Si los efectos incluyen llamadas a otro sistema, registra trabajos pendientes junto con el cursor en una transacción local. Un trabajador los envía usando una clave idempotente por cambio. Guardar el cursor antes de registrar el trabajo puede perder actualizaciones.

Cómo tratar una eliminación

operation=delete contiene identifier y version con data=null. Retira el recurso de la copia activa y conserva una marca local de eliminación con su versión. Así, un aviso antiguo no puede recrearlo. Una desactivación es distinta: llega como update con status=inactive y mantiene la cuenta.

Si el cursor ha caducado

HTTP/1.1 410 Gone
Content-Type: application/json

```
{"error":{"code":"cursor_expired",  
  "message":"El punto de lectura ya no está disponible.",  
  "requestId":"REQUEST-ejemplo","details":{}}}
```

Consulta changesRetentionSeconds en GET /connection. Tras un 410, crea otro checkpoint y reconstruye la copia mediante una carga completa y sus cambios posteriores. Mantén la copia anterior hasta completar la reconstrucción. No elimines registros por faltar en una página parcial; compara únicamente el conjunto completo del mismo ámbito autorizado.

Combinar el registro con los webhooks

Una estrategia sencilla es que el webhook solicite una lectura de cambios desde el cursor confirmado. Agrupa avisos y permite un solo consumidor por estado de sincronización; consulta también periódicamente para recuperar avisos que no hayan llegado. El webhook nunca proporciona ni avanza el cursor de cambios.

Si además aplicas el dato del webhook directamente, compara su versión con la copia local. Cuando el aviso no incluya versión o todos los campos necesarios, consulta el recurso o consume el registro de cambios. Recibir un webhook no autoriza a saltarte un lote pendiente.

Actividades, inscripciones y contenido

Consultar y gestionar eventos, participantes y publicaciones con sus estados y reglas de negocio.

En este bloque

Consultar y crear actividades

Cancelar una inscripción

Editar, publicar y cancelar una actividad

Consultar, crear y editar publicaciones

Inscribir y consultar participantes

Consultar y crear actividades

```
GET {{FEENDING_BASE_URL}}/ext/v2/events/{{EVENT_ID}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

```
GET {{FEENDING_BASE_URL}}/ext/v2/events?from={{FROM_UTC}}&to={{TO_UTC}}&limit={{PAGE_SIZE}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

from/to seleccionan actividades que solapan el intervalo. La consulta individual devuelve ETag; el listado es paginado y respeta la visibilidad de la conexión.

```
POST {{FEENDING_BASE_URL}}/ext/v2/events
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
```

```
{
  "title": "Encuentro de la comunidad",
  "description": "Presentación de proyectos",
  "starts": "2026-11-10T09:00:00Z", "ends": "2026-11-10T11:00:00Z",
  "timeZone": "Europe/Madrid", "visibility": "public"
}
```

title, starts, ends y timeZone son obligatorios. ends debe ser posterior a starts. description es anulable. visibility admite public o restricted; si se omite, se utiliza restricted. El evento se crea en estado draft.

```
HTTP/1.1 201 Created
Location: /ext/v2/events/EVENT-A
ETag: "1"
Content-Type: application/json
```

```
{"data":{"identifier":"EVENT-A","title":"Encuentro de la comunidad",
  "status":"draft","visibility":"public","version":1}}
```

Requiere events:write. Genera event-created, sin invitar ni cobrar automáticamente a participantes.

Editar, publicar y cancelar una actividad

```
PATCH {{FEENDING_BASE_URL}}/ext/v2/events/{{EVENT_ID}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
If-Match: {{RESOURCE_ETAG}}
```

```
{"starts": "2026-11-10T10:00:00Z", "ends": "2026-11-10T12:00:00Z"}
```

Admite title, description, starts, ends, timeZone y visibility. Devuelve el evento con nuevo ETag y genera event.changed. Se aplican las reglas del evento; no se modifican pagos o tipos de entrada enviando campos arbitrarios.

Publicar

```
POST {{FEENDING_BASE_URL}}/ext/v2/events/{{EVENT_ID}}/publish
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
If-Match: {{RESOURCE_ETAG}}
```

```
{}
```

Responde 200 con el evento publicado, después de comprobar permisos y datos.

Cancelar

```
POST {{FEENDING_BASE_URL}}/ext/v2/events/{{EVENT_ID}}/cancel
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
If-Match: {{RESOURCE_ETAG}}
```

```
{"reason": "Actividad aplazada"}
```

Responde 202 con data.operationIdentifier. Consulta GET /operations/{operationIdentifier} hasta obtener status=succeeded o failed. La respuesta incluye resourceIdentifier y un error estructurado si falla.

La cancelación conserva el historial y aplica las reglas de avisos y devoluciones del evento. Un timeout se recupera con la misma clave de operación. No sustituya la cancelación por un borrado de contenido.

Inscribir y consultar participantes

```
GET {{FEENDING_BASE_URL}}/ext/v2/events/{{EVENT_ID}}/registrations?limit={{PAGE_SIZE}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

```
GET {{FEENDING_BASE_URL}}/ext/v2/registrations/{{REGISTRATION_ID}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

Requiere registrations:read. La inscripción individual incluye ETag, userIdentifier, eventIdentifier, status y paymentStatus.

```
POST {{FEENDING_BASE_URL}}/ext/v2/events/{{EVENT_ID}}/registrations
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
```

```
{"userIdentifier": "USER-A", "quantity": 1}
```

userIdentifier es obligatorio. quantity es un entero positivo; se toma una plaza si se omite. Si hay varios tipos de entrada, envía ticketIdentifier obtenido de GET /events/{identifier}/tickets. Feending comprueba aforo, invitaciones, exclusividad y condiciones de compra.

```
HTTP/1.1 201 Created
Location: /ext/v2/registrations/REGISTRATION-A
ETag: "1"
Content-Type: application/json
```

```
{"data": {"identifier": "REGISTRATION-A", "userIdentifier": "USER-A",
  "eventIdentifier": "EVENT-A", "status": "pending_payment",
  "paymentStatus": "pending", "version": 1,
  "checkoutUrl": "<enlace-de-pago>"}}
```

Una inscripción gratuita aceptada devuelve status=confirmed y omite checkoutUrl. Si requiere pago, redirige al usuario al checkout y espera confirmación: 201 no demuestra que haya pagado. No fabriques referencias de pago.

Cancelar una inscripción

```
POST {{FEENDING_BASE_URL}}/ext/v2/registrations/{{REGISTRATION_ID}}/cancel
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
If-Match: {{RESOURCE_ETAG}}
```

```
{"reason": "El participante no puede asistir"}
```

Requiere registrations:write. Devuelve 202 con operationIdentifier. Consulta GET /operations/{identifier} y vuelve a leer la inscripción para conocer el resultado final. Se conserva su historial y se aplican las reglas de devolución de la actividad.

Elemento	Cómo se utiliza
409 capacity_exceeded	No hay plazas para una nueva inscripción. No repitas la petición como si fuera un error de red.
403 invitation_required	La persona necesita una invitación válida. La clave no omite ese requisito.
409 already_registered	Existe una inscripción incompatible. Consulta la existente antes de continuar.
409 cancellation_not_allowed	La inscripción o el evento no permite cancelar en ese estado.
422 ticket_required / invalid_ticket	Selecciona una entrada válida del evento.

La solicitud genera registration.submitted; los cambios, registration.changed; una devolución registrada, registration.refunded. Son estados distintos del proceso y no todos implican un cobro confirmado.

Consultar, crear y editar publicaciones

```
GET {{FEENDING_BASE_URL}}/ext/v2/posts?status=published&limit={{PAGE_SIZE}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

```
GET {{FEENDING_BASE_URL}}/ext/v2/posts/{{POST_ID}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

```
POST {{FEENDING_BASE_URL}}/ext/v2/posts
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
```

```
{"title":"Nueva actividad","content":"Información para la comunidad"}
```

title y content son obligatorios. Devuelve 201 con identifier, status=draft, version, Location y ETag. Valida y sanitiza el contenido. Genera post-created.

```
PATCH {{FEENDING_BASE_URL}}/ext/v2/posts/{{POST_ID}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
If-Match: {{RESOURCE_ETAG}}
```

```
{"title":"Nueva fecha para la actividad"}
```

Admite title, content y description; description permite null. Responde 200 con nuevo ETag y genera post-updated. El PATCH no publica el contenido.

```
POST {{FEENDING_BASE_URL}}/ext/v2/posts/{{POST_ID}}/publish
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
If-Match: {{RESOURCE_ETAG}}
```

```
{}
```

Responde 200 con status=published y genera post-published. Requiere posts:write y acceso al recurso. Los cambios aparecen también en el registro de posts.

Webhooks y catálogo de eventos

Configurar receptores, verificar firmas, recuperar entregas y consultar los eventos y sus datos.

En este bloque

Configurar webhooks desde tu aplicación

Cambiar, probar y rotar un webhook

Consultar y recuperar entregas

Entrega nativa: petición HTTP y envelope

Entrega nativa: verificar la firma

Verificador de firma: código Python (1/2)

Verificador de firma: código Python (2/2)

Eventos de las nuevas operaciones

Webhooks de usuarios: altas y perfiles

Webhooks de usuarios: estados y suscripciones

Webhooks de asistencia y pagos

Ejemplos de pagos y fin de prueba

Webhooks de actividades e inscripciones

Webhooks de calendarios y suscripciones

Formato de los eventos de cambios

Avisos de conectores y disponibilidad

Identificar el formato antes de procesar

Privacidad: qué significa cada campo

Ejemplos: exclusivo y visible en comunidad

Campos ausentes y decisión del receptor

Altas de contenido y condiciones de envío

Avisos operativos: feedback, matches y errores

Pruebas de privacidad antes de activar

Recibir contenido con sus traducciones

Traducciones: límites y validación

Configurar webhooks desde tu aplicación

```
GET {{FEENDING_BASE_URL}}/ext/v2/webhook-events
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

```
{
  "data": [
    {
      "name": "user-updated",
      "schemaVersion": 1,
      "resource": "users",
      "payloadSchema": "user"
    }
  ],
  "nextCursor": null,
  "hasMore": false
}
```

Enumera los eventos disponibles para tu conexión. El catálogo de esta guía explica sus datos y condiciones; no incluye nombres reservados en la respuesta de eventos disponibles.

```
POST {{FEENDING_BASE_URL}}/ext/v2/webhooks
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
```

```
{
  "url": "https://integraciones.example.test/feending",
  "events": ["user-created", "user-updated", "user-deactivated"],
  "active": true
}
```

```
HTTP/1.1 201 Created
ETag: "1"
Content-Type: application/json
```

```
{
  "data": {
    "identifier": "WEBHOOK-A",
    "active": true,
    "version": 1,
    "signingSecret": "whsec_<secreto-de-firma>"
  }
}
```

Guarda signingSecret en el gestor de secretos. Se devuelve al crear; las consultas ordinarias lo omiten. La recuperación de la respuesta idempotente exige el mismo permiso y solo se conserva durante su ventana de validez.

Requiere webhooks:write. La URL debe ser HTTPS pública. Se rechazan destinos privados, credenciales en la URL y redirecciones. Solo puedes gestionar receptores de tu plataforma y suscribirte a eventos autorizados.

Cambiar, probar y rotar un webhook

```
GET {{FEENDING_BASE_URL}}/ext/v2/webhooks?limit={{PAGE_SIZE}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

```
GET {{FEENDING_BASE_URL}}/ext/v2/webhooks/{{WEBHOOK_ID}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

```
PATCH {{FEENDING_BASE_URL}}/ext/v2/webhooks/{{WEBHOOK_ID}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
If-Match: {{RESOURCE_ETAG}}
```

```
{"active":false}
```

El PATCH admite url, events y active. Pausar detiene nuevos intentos; las entregas pendientes conservan su historial para su recuperación.

Enviar una prueba

```
POST {{FEENDING_BASE_URL}}/ext/v2/webhooks/{{WEBHOOK_ID}}/test
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
```

```
{"event":"user-updated"}
```

Responde 202 con deliveryIdentifier. El mensaje está firmado e incluye test=true y datos ficticios. Confirma su recepción sin modificar registros reales.

Rotar el secreto

```
POST {{FEENDING_BASE_URL}}/ext/v2/webhooks/{{WEBHOOK_ID}}/rotate-secret
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
If-Match: {{RESOURCE_ETAG}}
```

```
{}
```

Responde 200 con signingSecret, previousSecretExpiresAt y nuevo ETag. Actualiza tu receptor antes de la fecha indicada. Durante la transición recibe firmas para ambas claves.

Consultar y recuperar entregas

```
GET {{FEENDING_BASE_URL}}/ext/v2/webhooks/{{WEBHOOK_ID}}/deliveries?limit={{PAGE_SIZE}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

```
GET {{FEENDING_BASE_URL}}/ext/v2/webhooks/{{WEBHOOK_ID}}/deliveries/{{DELIVERY_ID}}
api-key: {{FEENDING_API_KEY}}
Accept: application/json
```

```
{
  "data": {
    "identifier": "DELIVERY-A", "eventIdentifier": "NOTIFICATION-A",
    "eventType": "user-updated", "status": "retrying", "attempts": 2,
    "lastStatusCode": 503, "errorCode": "receiver_unavailable",
    "lastAttemptAt": "2026-10-02T10:00:00Z",
    "nextAttemptAt": "2026-10-02T10:01:00Z"
  }
}
```

status admite pending, delivering, retrying, delivered, rejected y exhausted. delivered significa que el receptor confirmó la recepción; no demuestra que haya terminado su trabajo interno.

```
POST {{FEENDING_BASE_URL}}/ext/v2/webhooks/{{WEBHOOK_ID}}/deliveries/{{DELIVERY_ID}}/retry
api-key: {{FEENDING_API_KEY}}
Accept: application/json
Content-Type: application/json
Idempotency-Key: {{OPERATION_ID}}
```

```
{}
```

Responde 202 y conserva el ID del evento y su contenido. Un receptor pausado o una entrega no recuperable devuelve 409 delivery_not_retryable. No se reenvían entregas ya confirmadas.

2xx confirma la recepción. Timeouts, 408, 429 y 5xx permiten reintentos; los demás estados rechazan la entrega. El receptor valida la firma sobre el cuerpo original y guarda el evento antes de confirmar.

Entrega nativa: petición HTTP y envelope

El servidor envía POST a tu URL con Content-Type: application/json. El cuerpo se firma exactamente como se transmite.

```
webhook-id: <id-del-evento>
webhook-timestamp: <segundos-unix-del-intento>
webhook-signature: v1,<hmac-sha256-en-base64>

{
  "id": "cce4d5f99dfd4f70b885d9a758b43446",
  "schemaVersion": 1,
  "type": "user-updated",
  "occurredAt": "2026-10-02T10:15:00+00:00",
  "platformId": 42,
  "platform": "feending-ejemplo",
  "subject": {"table": null, "identifier": null},
  "dataType": "userbaseext",
  "data": {
    "identifier": "USER-identificador-devuelto",
    "firstName": "Ana",
    "updated": "2026-10-02T10:15:00Z"
  }
}
```

Campo / cabecera	Semántica
id / webhook-id	ID estable del evento en reintentos; utilízalo para deduplicar en el ámbito de plataforma y consumidor.
schemaVersion	Versión de envelope: 1. Verificala antes de aplicar el evento.
type	Nombre literal de evento al que se suscribe tu receptor.
occurredAt	Fecha en que se preparó el mensaje para su envío. No la utilices como fecha exacta del cambio ni como número de versión del registro.
platformId / platform	Ámbito de origen. Compáralo con la configuración autorizada.
subject / dataType	Pueden no aportar referencia. dataType es informativo; no es un esquema estable para despachar eventos.
data	Payload del evento, objeto, array o null según emisor. Usa el esquema del evento suscrito; el envelope no define todos sus campos.
webhook-timestamp	Marca del intento de entrega. Puede cambiar en un reintento aunque el ID y el cuerpo se conserven.

Entrega nativa: verificar la firma

El secreto tiene formato `whsec_<base64>`. El material HMAC son los bytes decodificados después del prefijo; no el texto completo del secreto.

```
signed = UTF8(webhook_id + "." + webhook_timestamp + ".") + raw_body
signature = Base64(HMAC_SHA256(decoded_secret, signed))
header = "v1," + signature
```

Conserva el cuerpo HTTP original. Parsearlo y volver a serializarlo puede cambiar espacios, Unicode u orden de propiedades y romper la firma. Durante una rotación pueden llegar dos firmas separadas por un espacio; acepta cualquiera que coincida con una clave vigente.

Verificador Python ejecutable

El archivo verificador completo incluido a continuación incluye comparación en tiempo constante, validación de timestamp, UUID y múltiples firmas. Ejemplo de uso en tu manejador HTTP:

```
from time import time
from verify_native_webhook import verify_signature

event_id = verify_signature(
    raw_body,
    request_headers,
    signing_secret,
    now_seconds=int(time()),
    max_clock_skew_seconds=configured_clock_tolerance,
)
# Después de verificar: decodificar JSON y validar el envelope.
# Comprobar platformId, schemaVersion, type e id == event_id.
# Persistir en una inbox con unicidad antes de responder 2xx.
```

La tolerancia temporal es una política de tu receptor: configúrala y sincroniza su reloj. El verificador no persiste mensajes ni sustituye la lógica idempotente de tu servicio.

Vector de interoperabilidad

Vector público de prueba utilizado por el transporte C#; no es una credencial real.

```
secret: whsec_AAECaWQFBgcICQoLDA00DxAREhMUFYXGBkaGxwdHh8=
id: cce4d5f99dfd4f70b885d9a758b43446
timestamp: 1790892000
body UTF-8: {"type":"event.changed","title":"València"}
signature: v1,rJkSWZp0LWqM76rx+2BZh7K5p3T5aTIi0o85LG4KUJk=
```

El vector comprueba solo la firma; ese body no representa el envelope completo que debe validar un receptor de producción.

Verificador de firma: código Python (1/2)

Python 3.10 o posterior, biblioteca estándar. Copia los dos bloques consecutivos en el mismo módulo. Las constantes pertenecen al contrato del perfil nativo; la tolerancia temporal la proporciona el consumidor.

```
"""Signature verification for Feending native webhook delivery (schema v1).
```

```
Call with the exact HTTP body bytes before decoding JSON. The caller supplies
the current time and its configured clock tolerance; this module performs no I/O.
"""
```

```
import base64
import binascii
import hashlib
import hmac
from collections.abc import Mapping
from uuid import UUID
```

```
SECRET_PREFIX = 'whsec_'
SIGNATURE_VERSION = 'v1'
MINIMUM_SECRET_BYTES = 32
ID_HEADER = 'webhook-id'
TIMESTAMP_HEADER = 'webhook-timestamp'
SIGNATURE_HEADER = 'webhook-signature'
```

```
class InvalidWebhook(ValueError):
    """The request must not be accepted as an authenticated delivery."""
```

```
def verify_signature(body: bytes, headers: Mapping[str, str], secret: str,
                    *, now_seconds: int, max_clock_skew_seconds: int) -> str:
    """Return the authenticated webhook-id or raise InvalidWebhook.
```

```
A bad secret or clock policy raises ValueError as a configuration error.
Deduplication and durable acceptance remain the receiver's responsibility.
"""
```

```
if max_clock_skew_seconds < 0 or not secret.startswith(SECRET_PREFIX):
    raise ValueError('Invalid webhook verification configuration')
key = base64.b64decode(secret[len(SECRET_PREFIX):], validate=True)
if len(key) < MINIMUM_SECRET_BYTES:
    raise ValueError('Webhook signing secret is too short')
```

Verificador de firma: código Python (2/2)

Continuación de la función `verify_signature`. Se conserva la sangría del cuerpo de la función.

```
normalized = {name.lower(): value for name, value in headers.items()}
try:
    identifier = normalized[ID_HEADER]
    timestamp = normalized[TIMESTAMP_HEADER]
    candidates = normalized[SIGNATURE_HEADER].split()
    if UUID(identifier).int == 0 or not timestamp.isascii() or not timestamp.isdecimal():
        raise ValueError('Invalid identifier or timestamp')
    sent_at = int(timestamp)
except (KeyError, ValueError, AttributeError) as error:
    raise InvalidWebhook('Invalid signature headers') from error
if sent_at <= 0 or abs(now_seconds - sent_at) > max_clock_skew_seconds:
    raise InvalidWebhook('Webhook timestamp outside the accepted window')
signed = f'{identifier}.{timestamp}'.encode('utf-8') + body
expected = hmac.new(key, signed, hashlib.sha256).digest()
for candidate in candidates:
    version, separator, encoded = candidate.partition(',')
    if not separator or version != SIGNATURE_VERSION:
        continue
    try:
        supplied = base64.b64decode(encoded, validate=True)
    except (binascii.Error, ValueError):
        continue
    if hmac.compare_digest(expected, supplied):
        return identifier
raise InvalidWebhook('No matching webhook signature')
```

Tras verificar la firma, valida el envelope y persiste el evento de forma idempotente antes de confirmar su recepción. `InvalidWebhook` significa rechazo de autenticidad; un error de configuración del secreto debe diagnosticarse en el servidor.

Eventos de las nuevas operaciones

Estos eventos se generan después de confirmar un cambio efectivo. Repetir una petición idempotente no vuelve a emitirlos.

Elemento	Cómo se utiliza
user-created / user-updated	Alta mediante POST /users y actualización del perfil. data contiene el usuario y su version.
user-deactivated / user-reactivated	Cambio de status. data contiene identifier, status, version y updated.
group-member-added / group-member-removed	Cambio de pertenencia: groupIdentifier, userIdentifier, present y version.
event-created	Alta mediante la API v2: identifier, status y version. Es un resumen; no permite decidir privacidad por sí solo. El alta de agenda con objeto completo se describe en «Altas de contenido y condiciones de envío».
event.changed	Edición, publicación o cancelación confirmada. Utiliza el formato de cambios del catálogo.
registration.submitted / registration.changed / registration.refunded	Solicitud, cambios y devolución de una inscripción.
post-created / post-updated / post-published	Creación, edición y publicación. data incluye identifier, status y version. Estos resúmenes de la API v2 no son el objeto completo del aviso de alta de publicaciones de agenda.

Los eventos nuevos de usuario, grupo y publicación utilizan el envelope nativo: id, schemaVersion, type, occurredAt, platformId y data. La clave de deduplicación es el ID del evento; un mismo recurso puede cambiar muchas veces.

Webhooks de usuarios: altas y perfiles

Utiliza estos nombres exactos al seleccionar eventos. Los objetos de usuario pueden contener campos adicionales y omitir valores vacíos. En el perfil nativo, el objeto o array descrito se encuentra en data.

Evento	Cuándo se envía y qué contiene
user-created	Al crear una cuenta desde la API v2 o desde la gestión de usuarios, incluida la invitación. data contiene un objeto de usuario; utiliza identifier para localizarlo.
users-created	Al crear o importar varias cuentas desde las operaciones por lotes. data es un array de usuarios. Recorre cada elemento; no lo proceses como un objeto único.
user-registered	Al completarse el registro desde la aplicación Feending. Contiene el usuario registrado. El registro mediante POST /ext/v1/auth/register no emite este evento.
user-updated	Al guardar cambios en los perfiles mediante las operaciones que generan este aviso. Contiene el usuario actualizado; no presupongas que toda importación o cambio de datos relacionados lo emite.
user-deleted	Al eliminar una cuenta mediante las operaciones de eliminación que generan el aviso. Contiene los datos del usuario eliminado, incluido su identifier.
user-approved	Al aprobar la incorporación de un usuario. Contiene el usuario con su estado de aprobación actualizado.

Webhooks de usuarios: estados y suscripciones

Evento	Cuándo se envía y qué contiene
user-approved-undo	Al retirar una aprobación anterior. Contiene el usuario tras el cambio; no equivale a eliminar su cuenta.
user-rejected	Al rechazar la incorporación de un usuario. Contiene el usuario con su estado de rechazo actualizado.
user-rejected-undo	Al deshacer un rechazo. Contiene el usuario tras el cambio; no presupongas que eso lo aprueba automáticamente.
user-subscription-canceled	Cuando se procesa la notificación de eliminación de una suscripción de Stripe y se localiza al usuario asociado. Contiene el usuario, no el objeto de suscripción de Stripe.
user-subscription-expired	Cuando la actualización de una suscripción de Stripe confirma su terminación y se localiza al usuario asociado. Contiene el usuario. No es un aviso genérico de cualquier pago fallido.
user-trial-ended	Cuando se detecta que ha finalizado el periodo de prueba de un usuario en una plataforma que utiliza esa función. Identifica a la persona mediante <code>userIdentifier</code> ; el formato se muestra en la sección de ejemplos.

Una notificación repetida no debe volver a ejecutar el mismo efecto. Para entregas nativas, utiliza el ID del evento para deduplicar; para periodos de prueba, protege también la acción asociada al usuario y al periodo.

Webhooks de asistencia y pagos

Evento	Cuándo se envía y qué contiene
event-attendance-registration	Al registrar una inscripción en los procesos que generan este aviso: aplicación, inscripción mediante API y determinados procesos de pago o invitación. data.identifier es el usuario y data.eventIdentifier es la actividad.
event-attendance-confirmed	Al confirmar la asistencia mediante la acción correspondiente de la aplicación. Contiene el registro de asistencia. No confundas este evento con la inscripción previa.
event-attendance-rejected	Al retirar una inscripción en los procesos que generan el aviso; también aparece en determinados procesos de devolución. Contiene el registro de asistencia actualizado. Comprueba sus campos antes de atribuir una causa concreta.
payment-confirmed	Al confirmar determinados pagos de suscripción, incluidos los procesos de alta y renovación de Stripe que generan este aviso. data contiene la suscripción de Feending, con campos como code, userIdentifier, productCode y paymentProvider. No presupongas que representa todos los tipos de cobro.

Ejemplo de datos de una inscripción

Proyección de data en un evento event-attendance-registration:

```
{
  "identifier": "USER-identificador-devuelto",
  "eventIdentifier": "EVENT-identificador-devuelto",
  "email": "ana@example.test"
}
```

El usuario se llama identifier en este mensaje. En la respuesta de POST /ext/v1/events/attend aparece como userIdentifier. No intercambies los dos formatos.

Ejemplos de pagos y fin de prueba

payment-confirmed

Proyección de data. El ejemplo muestra una suscripción, no un recibo completo ni el mensaje original del proveedor de pagos.

```
{
  "code": "SUBSCRIPTION-codigo",
  "userIdentifier": "USER-identificador-devuelto",
  "productCode": "PRODUCT-codigo",
  "paymentProvider": "stripe"
}
```

user-trial-ended

Campos del mensaje de fin de prueba. Los valores son ficticios.

```
{
  "userId": 123,
  "userIdentifier": "USER-identificador-devuelto",
  "email": "ana@example.test",
  "platformId": 42,
  "platformName": "feending-ejemplo",
  "activatedAt": "2026-09-01T10:00:00Z",
  "trialDays": 14,
  "affectedRoles": ["trial"],
  "assignedRoles": ["premium"]
}
```

Los códigos de rol y la duración dependen de la plataforma. El receptor debe decidir qué acción ejecutar según su configuración; no asumas que los valores del ejemplo existen en tu integración.

Webhooks de actividades e inscripciones

Estos eventos notifican cambios mediante identificadores y un conjunto de campos. Se utilizan en conexiones con este perfil habilitado; su disponibilidad se consulta en GET /ext/v2/webhook-events. No sustituyen a los eventos de asistencia anteriores.

Evento	Cuándo se envía y qué contiene
event.changed	Alta, modificación o eliminación de una actividad. En una modificación se comprueban los campos notificados. data: identifier, updated, status, starts, ends, is_public, published.
registration.changed	Alta, modificación o eliminación de una solicitud de inscripción. data: identifier, event_identifier, version, quantity, admission_state, payment_state, entitlement_state, checked_in_quantity, credential_version.
registration.submitted	Cuando se presenta una solicitud de inscripción, al crearla ya presentada o al presentarla por primera vez. data: identifier, event_identifier, version, quantity, admission_state, payment_state, entitlement_state.
registration.ticket.changed	Cuando cambia la versión de la credencial de una inscripción. data: identifier, event_identifier, version, credential_version, quantity, entitlement_state, checked_in_quantity.
registration.payment.changed	Alta, eliminación o cambio relevante en el pago de una inscripción. data: identifier, registration_identifier, version, capture_required, refund_required, cancel_authorization_required, requires_review. También puede enviarse por cambios en importes o referencias de pago que no se incluyen en este resumen.
registration.refunded	Cuando aumenta el importe reembolsado de un pago de inscripción. data: identifier, registration_identifier, version. Este mensaje no incluye el importe devuelto.

Webhooks de calendarios y suscripciones

Evento	Cuándo se envía y qué contiene
calendar.contact.changed	Alta, modificación o eliminación de una relación de contacto del calendario. data: identifier, version, following.
calendar.proposal.changed	Alta, modificación o eliminación de una propuesta del calendario. data: identifier, version, state.
calendar.profile.changed	Alta, modificación o eliminación de un perfil del calendario. data: platform_id, version, state, accept_followers. La referencia de subject.identifier es la plataforma.
subscription.state.changed	Al aplicar una revisión del estado de una suscripción que modifica su estado o el permiso de acceso. data: subscription_code, applied_revision, status, access_allowed, checked_at.
subscription.terminated	Al registrar o modificar la terminación de una suscripción. data: subscription_code, terminated_at. La referencia de subject.identifier es el código de suscripción.

Los campos de data de estos eventos conservan los nombres con guion bajo mostrados en las tablas. No los conviertas automáticamente a camelCase al buscar sus valores.

Formato de los eventos de cambios

Ejemplo de event.changed para una actualización. Se muestra una proyección de los campos de la actividad:

```
{
  "id": "cce4d5f9-9dfd-4f70-b885-d9a758b43446",
  "schemaVersion": 1,
  "type": "event.changed",
  "occurredAt": "2026-10-02T10:15:00Z",
  "platformId": 42,
  "subject": {"table": "events", "identifier": "EVENT-ejemplo"},
  "operation": "update",
  "data": {
    "identifier": "EVENT-ejemplo",
    "starts": "2026-10-20T09:00:00Z"
  },
  "previousData": {
    "identifier": "EVENT-ejemplo",
    "starts": "2026-10-20T08:00:00Z"
  }
}
```

operation indica insert, update o delete. En una eliminación, data conserva los campos notificados del registro eliminado. previousData puede aportar los valores anteriores en una actualización; admite que no esté presente.

Este formato no requiere platform ni dataType. Comprueba platformId, type y schemaVersion, y valida el contenido según el evento. Utiliza la misma comprobación de firma del transporte nativo descrita más adelante.

Identificar el formato antes de procesar

El nombre event-created no basta para conocer el JSON: distingue el transporte y la operación de origen. Conserva un ejemplo real de tu conexión, sin datos personales, y valida su estructura antes de activar la publicación automática.

Contrato	Qué recibe el consumidor
Contenido de agenda	El alta de agenda aporta el objeto de evento: puede incluir identifier, title, isExclusive, isPublic, shareable, groupsCodes, published, children. Los ejemplos de privacidad de esta guía corresponden a este objeto.
Operaciones de API v2	El aviso de creación puede ser un resumen identifier, status, version. No contiene por ello toda la información de visibilidad. No lo trates como un evento completo.
Referencia event.changed	Utiliza los campos documentados del registro de cambios, incluido is_public. No renombres ese campo a isExclusive: representan conceptos distintos.

Entrega nativa

El nombre está en type y el contenido en data. Verifica las cabeceras de firma, schemaVersion y la comunidad según las secciones siguientes. La firma autentica el envío; no concede permiso para publicar cualquier dato recibido.

Entrega convencional mediante un intermediario

El intermediario puede transformar el cuerpo antes de entregarlo a tu URL. No supongas que incluye type, firma nativa o ID de entrega nativo. Feending debe facilitar el ejemplo HTTP del receptor configurado, la ubicación del nombre y del objeto de evento, la autenticación y la política de reintentos de esa conexión.

Si la conexión entrega rawData, es texto JSON: deserialízalo una vez. Si entrega un objeto data, úsalo como objeto. Elige la ruta conforme al contrato de tu conexión; no busques indistintamente campos de nombres parecidos.

Una respuesta HTTP 2xx confirma recepción, no publicación en WordPress. Guarda el resultado de ambos pasos por separado y deduplica antes de crear la entrada.

Privacidad: qué significa cada campo

En el objeto completo de un evento, **isExclusive: true** identifica acceso privado o exclusivo. shareable no es un indicador de privacidad. Un evento no exclusivo también puede tener audiencia restringida.

Campo	Interpretación
isExclusive	Booleano de acceso exclusivo. Si es true, no exportes el evento como anuncio público aunque permita compartir invitaciones.
shareable	Opción para compartir el evento. En eventos exclusivos permite compartir invitaciones cuando está activada. No equivale a evento público, no controla por sí sola el envío del webhook y no debe interpretarse como autorización para republicar.
isPublic	Visibilidad en la presentación pública de Feending. false no significa necesariamente exclusivo: puede ser un evento ordinario de la comunidad. true tampoco sustituye las comprobaciones de acceso, audiencia y publicación.
groupsCodes	Audiencia por grupos. Un código distinto de __all__ indica restricción. En el objeto completo, null o lista vacía no restringen por grupos; __all__ representa toda la comunidad, no todo Internet.
isPublished y published	Estado publicado y fecha de publicación. Una fecha futura no autoriza publicar todavía. No confundir published con la fecha de celebración.
requiresRegistrationApproval	Necesidad de aprobar inscripciones. No lo uses como sustituto de isExclusive.
children	Eventos incluidos dentro del padre. Comprueba también su elegibilidad; la privacidad del padre no garantiza la de los hijos.

No reconstruyas la privacidad a partir del título, del precio, de la ausencia de asistentes o de metadatos internos. Usa los campos del contrato correspondiente.

Ejemplos: exclusivo y visible en comunidad

Fragmentos ilustrativos del **objeto de evento**, no peticiones HTTP completas. Los valores están explicitados para comparar significados; un formato compacto puede omitir los valores predeterminados.

Privado aunque se puedan compartir invitaciones

```
{
  "identifier": "EVENT-EJEMPLO-PRIVADO",
  "isExclusive": true,
  "shareable": true,
  "isPublic": false,
  "isPublished": true,
  "published": "2026-10-01T09:00:00Z"
}
```

No crear una entrada pública. shareable: true no anula isExclusive: true. El filtro de contenido excluye este objeto del aviso de alta de agenda; sirve aquí para reconocerlo en datos históricos o en otros recorridos autorizados.

Ordinario de comunidad, sin presentación pública

```
{
  "identifier": "EVENT-EJEMPLO-COMUNIDAD",
  "isExclusive": false,
  "shareable": false,
  "isPublic": false,
  "groupsCodes": ["__all__"],
  "isPublished": true,
  "published": "2026-10-01T09:00:00Z",
  "children": []
}
```

No es exclusivo ni está limitado a un grupo concreto. Puede ser elegible para un webhook de contenido configurado aunque isPublic y shareable sean false. Publicarlo en otra web exige que esa sea la finalidad autorizada de la conexión. Si solo replicas la presentación pública de Feending, exige además isPublic: true.

Campos ausentes y decisión del receptor

Omitido no siempre significa desconocido

El objeto de agenda serializado en formato compacto omite valores predeterminados: un booleano no nullable con valor false puede no aparecer; los valores null también pueden omitirse. Por ello, la ausencia de isExclusive puede representar false **solo si has identificado ese contrato de objeto completo**.

En un resumen identifier, status, version, la ausencia de isExclusive significa que ese resumen no informa de exclusividad. Lo mismo ocurre con la ausencia de audiencia o de hijos. No apliques el valor por defecto del objeto completo a un resumen ni conviertas null, cadenas o números en booleanos.

Antes de publicar en una web

1. Valida origen, comunidad, transporte y esquema. Separa las pruebas de los mensajes reales y conserva una clave de deduplicación adecuada al contrato.
2. Identifica si el payload es un objeto de contenido completo o un resumen. Si es un resumen, consulta el detalle permitido por la conexión. La respuesta debe aportar las condiciones de acceso necesarias; obtener solo isPublic no acredita todas ellas.
3. Comprueba exclusividad, audiencia, publicación efectiva y contenido anidado. Si falta información necesaria, conserva la entrada en borrador o revisión con el motivo; no publiques automáticamente.
4. Aplica la finalidad de la conexión. Para replicar exclusivamente la presentación pública, exige isPublic: true. Para otro canal autorizado, utiliza sus condiciones sin convertir shareable en un permiso.
5. Guarda la correspondencia entre comunidad, identificador de Feending e identificador del destino. Reprocesar un aviso no debe duplicar entradas.

Cambios posteriores

event-created informa de un alta; no es un aviso universal de cambios de privacidad ni de retirada. Utiliza los cambios o las consultas disponibles en tu conexión para reconciliar la copia. Si deja de estar autorizada su publicación, retírala del canal público. La ausencia de nuevos webhooks no demuestra que el contenido siga siendo publicable.

Altas de contenido y condiciones de envío

Estas reglas corresponden a los avisos con objeto completo de contenido. Una suscripción activa no convierte borradores, grupos privados o eventos exclusivos en anuncios públicos. Las operaciones v2 que notifican un resumen conservan el contrato explicado en «Eventos de las nuevas operaciones».

Evento	Contenido y condiciones
event-created	Alta de agenda con el objeto de evento y sus hijos. Debe estar publicado, con fecha alcanzada, sin rechazo, sin exclusividad ni vinculación a mentorías y sin grupos restringidos. Si algún hijo incumple, se omite el payload completo; no se recorta silenciosamente.
post-created	Alta de publicación con su objeto: identificador, contenido y datos de estado/audiencia. Debe estar publicada y aprobada, con fecha alcanzada, sin rechazo, contexto, newsletter, roles restringidos ni grupos restringidos.
perk-created	Alta de ventaja con identifier, title, content, status, published, groupsCodes, entre otros campos del objeto. Debe estar publicada, con fecha alcanzada y sin grupos restringidos.
comment-created	Alta de comentario: identifier, modelIdentifier, authorIdentifier, replyIdentifier, content, status, published. Debe estar publicado, con fecha alcanzada y sin eliminar; su publicación o evento padre debe ser exportable conforme a las reglas anteriores.

En estos avisos, el filtro no exige isPublic de la presentación pública para contenido ordinario. No interpreta shareable como privacidad. El receptor debe aplicar también el destino autorizado de la conexión.

Crear un borrador no garantiza recibir posteriormente un aviso *-created al publicarlo. Suscríbete a las operaciones de publicación o cambio que ofrezca tu conexión y comprueba el catálogo disponible. No deduzcas un contrato de actualización a partir del nombre de un alta.

Avisos operativos: feedback, matches y errores

Estos nombres tienen emisión en los recorridos descritos. Son datos para una integración autorizada, no contenido para una web pública. No les apliques como permiso de publicación las reglas de los anuncios de agenda.

Evento	Cuándo se envía y qué contiene
feedback-created	Nueva valoración persistida. data contiene identifier, authorIdentifier, targetIdentifier, targetModel, category, visibility, anonymous, content, value, según los valores del objeto. Actualizar una valoración existente no es una nueva alta. Respeta anonimato y visibilidad; no publiques datos personales del payload.
match-created	Creación manual efectiva de una conexión entre participantes, con el objeto de match. No se emite por una actualización ni por devolver una pareja ya existente. Este recorrido no garantiza avisos de todos los generadores automáticos de conexiones.
payment-error	Cargo fallido Stripe asociado y registrado. Incluye identifier, provider, amount, currency, failureCode, failureMessage. No es un aviso genérico de cualquier error del checkout ni implica que todos los intentos fallidos generen un pago registrado.

Ejemplo de datos de un pago fallido

```
{
  "identifier": "ch_ejemplo",
  "provider": "stripe",
  "amount": 25.50,
  "currency": "EUR",
  "failureCode": "card_declined",
  "failureMessage": "Your card was declined."
}
```

amount está en unidades principales de la divisa: 25,50 EUR, no 2550 céntimos. El motivo procede del proveedor y puede variar. No se incluye el objeto de cargo completo, sus metadatos ni los datos del medio de pago. El ejemplo representa data, no el envelope.

Pruebas de privacidad antes de activar

Ejecuta estas comprobaciones con datos de prueba en el receptor. Los ejemplos privados son fixtures para validar el rechazo; no requieren exportar contenido privado real.

Caso	Resultado esperado
Exclusivo con shareable=true	No se publica en abierto. Compartir invitaciones no elimina la exclusividad.
No exclusivo, limitado a un grupo	No se publica en abierto aunque isPublic=true.
Ordinario con isPublic=false	No se clasifica como exclusivo. Se excluye si el destino replica únicamente la presentación pública.
Padre ordinario con hijo restringido	No se publica el payload completo ni se filtra solo el padre.
Borrador o publicación futura	No se publica antes del estado y la fecha autorizados.
Objeto compacto conocido sin isExclusive	Se aplica el false del contrato; todavía se comprueban las demás restricciones.
Resumen sin datos de privacidad	No se considera público por omisión. Se obtiene información suficiente o se deja en revisión.
Booleano recibido como cadena o null	Se rechaza o aparta por esquema inválido; no se convierte por coerción.
Mismo aviso recibido dos veces	Una sola entrada en el destino, sin efectos duplicados.
Evento ya copiado cuya visibilidad cambia	La reconciliación detecta el cambio y retira la copia pública cuando corresponde.

Comprueba por separado recepción HTTP, interpretación del payload y estado final de la entrada. Un 200 del receptor no prueba que la entrada sea pública, privada o se haya retirado.

Recibir contenido con sus traducciones

Las suscripciones de webhooks de plataforma con objeto completo admiten `includeTranslations`. Con `true`, los campos habituales permanecen y se añade `data.translations`. Con `false`, se conserva el payload habitual. El contrato de notificaciones por referencias de `/ext/v2` no cambia.

Configuración por suscripción

Activa «Incluir traducciones disponibles» en la suscripción del panel. Si la vista no está disponible para tu perfil, solicita la configuración a Feending. En la API de administración, `GET/POST/PUT v4/platforms/webhooks` y `PATCH v4/platforms/webhooks/{identifier}` exponen esta preferencia; no utilices las credenciales de `/ext/v2` para estas operaciones. Al crear, la omisión equivale a desactivado; al actualizar, la omisión conserva el valor guardado.

Ejemplo del contenido recibido

```
{
  "data": {
    "title": "Una noche de jazz",
    "translations": {
      "en": { "title": "A night of jazz" },
      "pt_BR": { "title": "Uma noite de jazz" }
    }
  }
}
```

Es un fragmento ilustrativo de `data`, no el sobre HTTP completo. Si tu contrato entrega `rawData`, deserializa ese texto JSON una vez: contiene la misma proyección del contenido.

Tratamiento en el receptor

Las claves de idioma siguen la normalización de Feending y los campos usan `camelCase`. Lee solo los idiomas y campos presentes. Una entidad traducible sin textos guardados recibe `translations: {}`; una entidad no traducible conserva su forma habitual. La integración de destino debe guardar y relacionar las versiones idiomáticas; recibirlas no crea por sí solo traducciones en WordPress.

Traducciones: límites y validación

La preferencia de traducciones afecta únicamente al objeto principal y a los campos de texto expuestos por su contrato y declarados traducibles. No sustituye los campos originales ni traduce el sobre del webhook.

1. No genera traducciones durante el envío ni completa idiomas ausentes.
2. No incorpora traducciones de entidades hijas o relacionadas.
3. No emite otro webhook al modificar posteriormente una traducción.
4. No modifica los conectores de proveedores ni los avisos por referencias de /ext/v2.
5. No cambia la elegibilidad, la privacidad ni el destino autorizado del contenido.

Comprobar la recepción

1. Utiliza una entidad de prueba con una traducción guardada y activa la opción en una suscripción.
2. Provoca el aviso de contenido correspondiente y comprueba el idioma y los campos de translations.
3. Comprueba que el sistema receptor conserva el contenido original y crea o actualiza la versión idiomática sin duplicar la entidad.
4. Desactiva la opción y comprueba que el siguiente aviso vuelve al contrato habitual.
5. Prueba una entidad traducible sin traducciones: el bloque estará vacío. No lo interpretes como orden de borrar versiones anteriores.

Verifica por separado la recepción HTTP y el resultado en el sistema de destino. Una respuesta 2xx del intermediario no acredita que WordPress haya creado la entrada ni sus traducciones.

Avisos de conectores y disponibilidad

unith

El conector Unith puede reenviar un mensaje de conversación con el nombre unith. El cuerpo conserva el mensaje recibido del proveedor; no tiene un esquema común de usuario o actividad. Solo corresponde a conexiones que utilizan ese servicio.

Confirmar los eventos de la conexión

Los nombres del catálogo no implican que todos estén disponibles para cualquier credencial o transporte. Consulta `GET /ext/v2/webhook-events` cuando uses la API v2 y valida una entrega del recorrido concreto. Los avisos convencionales requieren la suscripción y el contrato de su receptor.

Validación de la integración

Comprobar el flujo completo y los casos de error antes de dar la conexión por terminada.

En este bloque

Comprobar la integración completa

Comprobar la integración completa

1. **Permisos.** GET /connection identifica tu plataforma. Una clave de lectura recibe 403 al intentar escribir.
2. **Alta.** Repite el POST con la misma Idempotency-Key: devuelve la misma cuenta sin duplicar efectos.
3. **Edición.** Actualiza el perfil y guarda el nuevo ETag. Una escritura con el anterior responde 412.
4. **Grupos.** Repite una incorporación: debe existir una sola pertenencia.
5. **Acceso.** Autentica en tu portal y solicita un enlace. La persona debe entrar en su propia cuenta.
6. **Baja.** Desactiva al usuario y comprueba que no puede entrar ni utilizar su sesión anterior. Reactivarlo exige un nuevo acceso.
7. **Sincronización.** Recupera desde un cursor anterior. Repetir el lote conserva el mismo resultado.
8. **Webhooks.** Envía una prueba, comprueba su entrega y simula un 503. El reintento no debe duplicar efectos.
9. **Contenido e inscripciones.** Crear no debe publicar ni cobrar implícitamente. Un pago pendiente no se muestra como confirmado.