betterworks

Public API NextGen Customer Guide

NextGen

Updated April/May 2026

API Changes for NexGen

Overview

Betterworks has upgraded the Conversations API, Feedback API & Goals API to support our NextGen platform.

Most integrations will continue to work as expected. However, there are a few important updates around how records are identified and how some fields behave.

This guide explains what's changed and what actions (*if any*) you need to take.

API by Module	Page(s)
Conversations - Last Updated April 2026	3 - 5
Feedback - Last Updated April 2026	6 - 9
Goals - Last Updated May 2026	10 - 13

Conversations API

What's Changing with the Conversations API?

1. IDs: UUIDs are now the primary identifier

Previously, most records used numeric IDs (like 123).

With NextGen:

- Every record has a **UUID** (a long string like e98d4f17-...)
- Numeric IDs may still exist, but:
  - They can sometimes be **0**
  - They should **no longer be relied on as the main identifier**

👉 **What this means for you:**

- Use **UUIDs** to identify records in your integration
- Treat numeric IDs as optional

2. Some numeric IDs may be 0

In NextGen-created data, numeric ID fields may appear as 0.

👉 **What this means for you:**

- This is expected behavior
- Your integration should **not fail or error** when encountering id = 0

3. APIs now accept both UUID and numeric IDs

Where you pass identifiers (e.g., in URLs or filters), you can now use:

- UUIDs (recommended)
- Numeric IDs (legacy support)

👉 **What this means for you:**

- No immediate breaking change
- But **we recommend moving to UUIDs**

4. Some filters are no longer supported

The following filters are **not available in NextGen**:

- target_user
- target_manager
- target_employee
- current

👉 **What this means for you:**

- If you use these filters today, you'll need to update your integration
- Refer to updated API documentation for supported filters

Endpoint-Specific Notes

Templates APIs

Endpoints:

- /api/v1/conversations/templates/
- /api/v1/conversations/templates/{template_id}/

Updates:

- Numeric IDs may be 0
- Some fields (like groups) may now use UUIDs

👉 **Action:**

- Ensure your system:
  - Accepts UUIDs
  - Does not fail on id = 0

Conversations API

Conversations API

Endpoint:

- /api/v1/conversations/

Updates:

- Filters like deployment_id, manager_id, employee_id accept UUID or numeric ID
- Many nested objects may have id = 0
- UUID fields are available for all key entities

👉 **Action:**

- Use UUIDs wherever possible
- Make your integration tolerant of optional/missing numeric IDs

---

What You Need to Do

Required Updates

- ✅ Use **UUID as the primary identifier**
- ✅ Ensure your system **handles id = 0 gracefully**
- ✅ Accept both UUID and numeric ID in inputs

Recommended (Best Practice)

- Prefer UUIDs for all new integrations
- Avoid relying on numeric IDs for logic or joins

If You Do Nothing

Most integrations will continue working **unless**:

- You strictly require numeric IDs
- You assume IDs are always non-zero
- You rely on deprecated filters

Feedback API

What's Changing with the Feedback API?

1. IDs: UUIDs are now the primary identifier

Previously, most records used numeric IDs (like 123).

With NextGen:

- Every record has a **UUID** (a long string like e98d4f17-...)
- Numeric IDs may still exist, but:
  - They can sometimes be **0**
  - They should **no longer be relied on as the main identifier**

👉 **What this means for you:**

- Use **UUIDs** to identify records in your integration
- Treat numeric IDs as optional

2. Some numeric IDs may be 0

In NextGen-created data, numeric ID fields may appear as 0.

👉 **What this means for you:**

- This is expected behavior
- Your integration should **not fail or error** when encountering id = 0

Endpoint-Specific Notes

Feedback API

Endpoint:

- /api/v1/feedback/

Updates:

- Numeric IDs may be 0
- size is max capped at 1000

👉 **Action:**

- Ensure your system:
  - Accepts UUIDs
  - Does not fail on id = 0

Feedback Request API

Endpoint:

- /api/v1/feedback/requests

Updates:

- Numeric IDs may be 0
- feedback_id is always 0
- feedback_uuid is always null
- size is max capped at 1000
- status supports a new value “pending”. All possible values are
  - pending (new)
  - answering
  - submitted
  - declined
  - expired

👉 **Action:**

- Use UUIDs wherever possible
- Make your integration tolerant of optional/missing numeric IDs
- Replace feedback_id & feedback_uuid with id and uuid, respectively

Feedback API

Templates APIs

Endpoints:

- /api/v1/feedback/templates/
- /api/v1/feedback/templates/{template_id}/

Updates:

- Numeric IDs may be 0
- Summary_visible_employee will always be true
- summary_visible_manager will always be true

👉 **Action:**

- Use UUIDs wherever possible
- Make your integration tolerant of optional/missing numeric IDs

---

Participant APIs

Endpoints:

- /api/v1/feedback/templates/{template_id}/employees
- /api/v1/feedback/templates/{template_id}/providers

Updates:

- Numeric IDs may be 0

👉 **Action:**

- Ensure your system:
  - Accepts UUIDs
  - Does not fail on id = 0

What You Need to Do

Required Updates

- ✅ Use **UUID as the primary identifier**
- ✅ Ensure your system **handles id = 0 gracefully**
- ✅ Accept both UUID and numeric ID in inputs

Recommended (Best Practice)

- Prefer UUIDs for all new integrations
- Avoid relying on numeric IDs for logic or joins

If You Do Nothing

Most integrations will continue working **unless**:

- You strictly require numeric IDs
- You assume IDs are always non-zero
- You rely on deprecated filters

Goals API

What's Changing with the Goals API?

IDs: UUIDs are now the primary identifier

1. The uuid field is the primary identifier for all entities

2. (⚠️) The numeric id field is 0 for all nextgen-created entities in API responses.

(❓) When will I receive numeric IDs?

- For all entities such as goals, categories, assessments etc. returned in the API responses which are created via the NextGen public APIs ***the numeric core IDs will be 0.***
- If the entities have been migrated from the legacy data ***the core ID will be present.***

3. (⛔) The URL path params and query params will ONLY take UUIDs, NOT numeric IDs

For instance, the GET /api/v1/goals/{goal_id}/ endpoint will **always take a UUID, not the numeric ID.**
(❓) Why do we not support numeric IDs in params for backwards compatibility?

- Goals NextGen entities are not created with a numeric ID.
- We'd have to support search via created_context.core_id which would lead to more indexes and compromise on performance.
- If a customer migrates from legacy and then creates new goals/categories in NextGen, there'd be data that has numeric IDs as well as UUIDs, and we'd need to support search over both in the same request, which makes little sense.

4. (⚠️) Views are not supported.

- `stats.views` will always be 0.
- `viewed_by_owner_date` filter not supported

Endpoint-Specific Notes

Goals API

Endpoint:

- POST /api/v1/goals/
- PUT /api/v1/goals/{goal_id}/
- DELETE /api/v1/goals/{goal_id}/

Updates:

- **`visibility` only accepts `public` or `private`:** The legacy API documents `visibility` as accepting `public`, `locked`, or `private`. NextGen only accepts `public` or `private`. Sending `locked` returns a `400 Bad Request`. This is because goal locks have not been implemented yet in goals NextGen. NextGen stores and returns `downline` in place of `locked`. A goal with `visibility: "locked"` in the legacy system will be returned as `visibility: "downline"` from NextGen.
- **`creation_source` accepted values differ:** The legacy API documents `creation_source` as accepting `manual`, `converted`, or `default`. NextGen accepts `manual`, `csv_upload`, or `v1_api`. Sending `converted` or `default` returns a `400 Bad Request`.
- **`name` is validated to a maximum of 750 characters:** The legacy API does not document a maximum length for `name`. NextGen enforces a 750-character maximum. Exceeding this returns a `400 Bad Request`. This is due to the limitation being enforced in NextGen goals.
- **Key result `delete` and `create` flags are not supported in the PUT API:** The legacy API has `key_results[].delete` (boolean) and `key_results[].create` (boolean) flags for controlling key result lifecycle within an update request, which NextGen does not support. There is no way to delete a key result via this endpoint in NextGen, as a KR is treated as a goal in itself, and would require the DELETE endpoint to be deleted.
- **Response status code is `204 No Content` instead of `200 OK` in the DELETE API**

Actions:

- Ensure you can handle the updated creation_source values
- Ensure that the goal name is <750 characters.
- Ensure visibility is not set to locked until the relevant feature is implemented.
- Ensure that key results are created/deleted via separate calls using the same goals API.

Goals API

Assessment API

Endpoint:

- GET /api/v1/goals/{goal_id}/assessments/

Updates:

- **`goal_id` in response is a UUID string, not an integer:** As mentioned in the general changes, NextGen goals does not generate numeric IDs.

Actions:

- Ensure you can handle goal_id as UUIDs and not integers.

Comment API

Endpoint:

- GET /api/v1/goals/{goal_id}/comments/

Updates:

- **`goal_id` path parameter only accepts a UUID:** As mentioned in general changes.
- **`Comment.id` is a UUID string, not an integer:** As mentioned in general changes.

Actions:

- Ensure you can handle goal_id and comment.id as UUIDs and not integers.

Categories API

Endpoint:

- GET /api/v1/goals/categories/search/

Updates:

- **Category `id` is a UUID string, not an integer -** As mentioned in general changes.
- **`updated_by_id` and `created_by_id` are UUID strings, not integers**

Actions:

- Ensure you can handle category id, and the userID fields UUIDs and not integers.

Goals API

Goals Filter API

Endpoint:

- GET /api/v1/goals/filter/

Updates:

- **Some filter values are not supported. They'll still be accepted without 400 but will not affect the result. They are the following:**
- **`following`** | Filter goals the current user is following (since the 'goal following' feature is not implemented in NextGen goals)
- **`team_only`** | Filter goals owned by team members
- **`permission`** | Filter by `can_edit` / `can_only_view` (since permissions are controlled via policies in NextGen and we can't support filters based on that)
- **`viewed_by_owner_date`** | Filter by when owner viewed the goal (since views are deprecated)
- **`integration` filter accepts a different set of values:** The legacy API documents `integration` as accepting `jira`, `salesforce`, or `tableau`. NextGen accepts: `asana`, `azure_devops`, `docebo`, `excel365`, `github`, `google_sheets`, `jira`, `linkedinlearning`, `ms_teams`, `salesforce`, `slack`, `udemy`. `tableau` is not supported; sending it returns a `400 Bad Request`.
- **`aligned_to_owner` prefixed IDs are not handled:** The legacy API documents `aligned_to_owner` as accepting prefixed IDs like `u123`, `d456`, `t789`, `f012`. NextGen passes the value as-is to the filter and does not parse or expand these prefixes, since NextGen filters work on UUIDs.

Actions:

- Do not use deprecated filter values which are not supported.
- Ensure correct values for the integration filter.
- Ensure that UUIDs are passed to aligned_to_owner filter.