

DEXPRO AWF External Approvals — API Integration Guide

“ **Audience:** Developers building custom approval interfaces or integrations that consume the DEXPRO AWF External Approvals API directly.

Looking for the SharePoint / Teams reference implementation? See [SETUP-GUIDE.md](#) (admin setup) and [SHAREPOINT-CONTRACT.md](#) (SharePoint list contract).

Overview

The External Approvals API exposes pending approval entries directly as OData resources inside Business Central's standard API V2 framework. Any system that can make HTTPS calls — a custom portal, a mobile app, a third-party notification service, an Azure Function — can:

- Poll for approvals assigned to specific people or groups
- Display record context to the approver
- Submit approve / reject decisions with an optional comment
- Handle group (claim-based) approvals

No SharePoint setup required. The API works independently of the SharePoint integration toggle (`SP Approvals Enabled`). As long as the DEXPRO AWF extension is installed and external approvers are configured in BC, the API pages are live.

When to use the API vs. the SharePoint integration

	API Integration	SharePoint Integration
Setup	None beyond BC API access	App registration, SP site, wizard
Notification	Your system handles it	Power Automate → Teams Adaptive Card
Response collection	Your UI calls the BC API directly	Approver responds in Teams; PA writes to SP; BC polls SP

	API Integration	SharePoint Integration
Licence needed (approver)	Paid BC user licence (see licensing note)	Paid BC user licence, plus M365 for Teams (see licensing note)
Good for	Custom portals, mobile apps, vendor self-service	Quick Teams rollout, minimal dev effort

“ **Licensing — customer/partner responsibility:** Acting on Business Central data — whether directly via the API or indirectly through the SharePoint relay — requires each approver to hold an **appropriate paid Business Central user licence**. A Microsoft 365 licence alone does **not** grant API or write access to BC, and routing actions through a service account does **not** remove the per-user requirement (Microsoft's multiplexing / indirect-access terms). A **Team Member** licence may be sufficient for approval-only use, but Microsoft restricts Team Member to designated scenarios and this is **not guaranteed** for custom or third-party interfaces — a full **Essentials/Premium** user may be required. **DEXPRO makes no licensing representation.** The customer and their Microsoft licensing partner are solely responsible for determining and maintaining correct licensing; verify against the current [Microsoft Dynamics 365 Licensing Guide](#).

Prerequisites

Requirement	Details
Business Central	DEXPRO AWF extension installed, BC 25 or later
Entra App Registration	An Entra ID app with <code>Financials.ReadWrite.All</code> (or a delegated user flow) for BC API access
BC permission set	The API caller's service account needs a permission set that grants read access to <code>DXP AWF Ext. Approval Entry</code> and write access for the bound actions (<code>approve</code> , <code>reject</code> , <code>claim</code> , <code>releaseClaim</code> , <code>markNotified</code> , <code>setTeamsMessageId</code>). The DXP AWF Admin set covers this. No explicit Execute permission on the API page is required — it is granted automatically via the page's inherent entitlements.
External Approvers	At least one external approver configured in BC (AWF Setup → External Approvers)

Authentication

The BC API V2 uses OAuth 2.0 with Microsoft Entra ID. Obtain a bearer token from:

```
POST https://login.microsoftonline.com/{tenantId}/oauth2/v2.0/token
```

```
grant_type=client_credentials
&client_id={clientId}
&client_secret={clientSecret}
&scope=https://api.businesscentral.dynamics.com/.default
```

Include the token on every request:

```
Authorization: Bearer {access_token}
```

🔑 **Note:** Client Credentials flow is suitable for backend service integrations. For user-delegated flows (e.g. a web portal where each approver authenticates as themselves), use Authorization Code + PKCE instead — the token then carries the user's identity, which is useful for audit purposes.

Base URL

```
https://api.businesscentral.dynamics.com/v2.0/{tenantId}/{environmentName}/api/dexpro/advanced
Workflow/v1.0
```

All examples below use `{baseUrl}` as a shorthand for this full base URL.

To find your company ID (a GUID required in most requests):

```
GET {baseUrl}/companies
```

Response (abbreviated):

```
{
  "value": [
    {
      "id": "5e9f4c3a-0001-ef11-9f8a-6045bd028c9f",
      "name": "CRONUS International Ltd.",
      ...
    }
  ]
}
```

Use the `id` value as `{companyId}` in subsequent requests.

Entities

Three read-only entity sets are available. All use `SystemId` (a platform-assigned GUID, exposed as `id`) as the OData key — the standard BC API V2 convention and a requirement for Power Automate / Power Apps compatibility.

Entity set	OData URL segment	OData key field	Description
Approval entries	<code>externalApprovalEntries</code>	<code>id</code> (SystemId)	The actionable entries — one per approver per approval request. Main entity for integrations.
Approvers	<code>externalApprovers</code>	<code>id</code> (SystemId)	The registered external approver records. Useful for bootstrap / pre-population.
Group members	<code>externalApproverMembers</code>	<code>id</code> (SystemId)	Members of external approver groups.
Entry document	<code>externalApprovalEntryDocuments</code>	<code>id</code> (SystemId)	Source document record as JSON. Separate endpoint — only fetched on demand.
Entry attachments	<code>externalApprovalEntryAttachments</code>	<code>id</code> (SystemId)	Standard BC document attachments for the source record. Separate endpoint — only fetched on demand.

“ **Why GUIDs, not integers?** The `entryNo` integer field is still present in the response body for display and correlation, but it is not the OData key. Using SystemId (GUID) as the key follows Microsoft's official BC API guidance and ensures compatibility with Power Automate, Power Apps, and Logic Apps connectors. Always use `id` in URL paths.

Working with Approval Entries

List all pending entries for one approver

```
GET {baseUrl}/companies({companyId})/externalApprovalEntries
?$filter=approverEmail eq 'john.doe@contoso.com' and status eq 'Pending'
&$orderby=createdDateTime asc
```

Response:

```
{
  "@odata.context": "...",
  "value": [
    {
      "id": "b9f3a2c1-0001-ef11-bf8d-6045bd028c9f",
      "entryNo": 42,
      "approvalEntryNo": 17,
      "workflowInstanceId": "a1b2c3d4-0000-0000-0000-000000000001",
      "templateCode": "PURCHASE-APPROVAL",
      "stageCode": "EXT-REVIEW",
      "stageDescription": "External Review",
      "documentNo": "PO-00123",
      "documentType": "Order",
      "approverEmail": "john.doe@contoso.com",
      "approverDisplayName": "John Doe",
      "status": "Pending",
      "isGroupApproval": false,
      "isRelatedApproval": false,
      "description": "Fabrikam Inc.",
      "recordDescription": "Purchase Order PO-00123",
      "amount": 15000.00,
      "dueDate": "2026-05-15",
      "senderUserId": "PROCUREMENT",
      "languageCode": "ENU",
      "processed": false,
      "createdDateTime": "2026-05-04T09:12:33Z",
      "errorMessage": "",
      "spListItemId": "",
      "spAttachmentsUrl": ""
    }
  ]
}
```

Retrieve a single entry

```
GET {baseUrl}/companies({companyId})/externalApprovalEntries(b9f3a2c1-0001-ef11-bf8d-6045bd028c9f)
```

Common filter patterns

```
# All pending entries for an approver (individual + group)
$filter=approverEmail eq 'jane@contoso.com' and processed eq false and status ne 'Cancelled'

# All pending entries for a group (show to all members before one claims)
$filter=extApproverGroupCode eq 'FINANCE-GROUP' and status eq 'Pending'

# Entries with processing errors (for admin monitoring)
$filter=errorMessage ne '' and processed eq false

# All actionable entries for a group member (pending or claimed by them)
$filter=approverEmail eq 'jane@contoso.com'
    and (status eq 'Pending' or status eq 'Notified' or status eq 'Claimed')
    and processed eq false
```

Bound Actions

Actions are called as OData bound actions via `POST`. The URL pattern is:

```
POST {baseUrl}/companies({companyId})/externalApprovalEntries({id})/Microsoft.NAV.{actionName}
Content-Type: application/json
```

`{id}` is the `id` field (SystemId GUID) from the entry. A successful call returns `200 OK` with the updated entry in the response body.

approve

Records an approval decision and advances the BC workflow.

```
POST {baseUrl}/companies({companyId})/externalApprovalEntries(b9f3a2c1-0001-ef11-bf8d-6045bd028c9f)/Microsoft.NAV.approve
Content-Type: application/json

{
  "approverEmail": "john.doe@contoso.com",
  "comment": "Looks good. Approved as per Q2 budget."
}
```

`approverEmail` is **required** — pass the email of the user who actually performed the approval. This is recorded in the BC audit log and on the approval entry. Since the API is typically called by a backend service (client credentials), BC has no other way to know the real actor's identity.

`comment` is optional. Omit or pass `""` to approve without a comment.

Response `200 OK`:

```
{
  "id": "b9f3a2c1-0001-ef11-bf8d-6045bd028c9f",
  "entryNo": 42,
  "approverEmail": "john.doe@contoso.com",
  "status": "Approved",
  "responseComment": "Looks good. Approved as per Q2 budget.",
  "responseDateTime": "2026-05-04T10:35:12Z",
  "processed": true,
  "processedDateTime": "2026-05-04T10:35:12Z",
  ...
}
```

reject

Records a rejection decision and drives the BC rejection flow (including reassignment for related approvals).

```
POST {baseUrl}/companies({companyId})/externalApprovalEntries(b9f3a2c1-0001-ef11-bf8d-6045bd028c9f)/Microsoft.NAV.reject
Content-Type: application/json
```

```
{
  "approverEmail": "john.doe@contoso.com",
  "comment": "Amount exceeds departmental authority. Escalate to CFO."
}
```

`approverEmail` is **required** for the same reasons as `approve`.

Response `200 OK`:

```
{
  "id": "b9f3a2c1-0001-ef11-bf8d-6045bd028c9f",
  "entryNo": 42,
  "approverEmail": "john.doe@contoso.com",
  "status": "Rejected",
  "responseComment": "Amount exceeds departmental authority. Escalate to CFO.",
  "responseDateTime": "2026-05-04T10:38:44Z",
  "processed": true,
}
```

```
...  
}
```

markNotified

Call this after your system has successfully delivered the approval notification to the approver (e.g. sent an email, shown the card in your portal). Transitions status from `Pending` → `Notified`. This is optional but recommended — it allows BC admins to distinguish between entries that were never delivered and entries awaiting a response.

```
POST {baseUrl}/companies({companyId})/externalApprovalEntries(b9f3a2c1-0001-ef11-bf8d-6045bd028c9f)/Microsoft.NAV.markNotified  
Content-Type: application/json  
  
{}
```

Response `200 OK`:

```
{  
  "id": "b9f3a2c1-0001-ef11-bf8d-6045bd028c9f",  
  "entryNo": 42,  
  "status": "Notified",  
  ...  
}
```

🔥 Only effective when current status is `Pending`. A no-op on any other status.

setTeamsMessageId

Call this **instead of** `markNotified` when your transport can later need to update/close the posted message (e.g. a Microsoft Teams adaptive card). It does everything `markNotified` does (Pending → Notified) **and** stores the transport's message identifier on the entry (BC field `Teams Message ID`) and mirrors it onto the SharePoint column `TeamsMessageId`. The Power Automate v2 flow uses this so a peer claim/decision can replace (close) other group members' still-open cards via *Update an adaptive card*.

```
POST {baseUrl}/companies({companyId})/externalApprovalEntries(b9f3a2c1-0001-ef11-bf8d-6045bd028c9f)/Microsoft.NAV.setTeamsMessageId  
Content-Type: application/json  
  
{ "teamsMessageId": "17000000000000" }
```

Response 200 OK:

```
{
  "id": "b9f3a2c1-0001-ef11-bf8d-6045bd028c9f",
  "entryNo": 42,
  "status": "Notified",
  "teamsMessageId": "17000000000000",
  ...
}
```

“ Pass the message identifier your transport returns when it posts the message — for the Teams connector's *Post card in a chat or channel* action that is `body/id`. Empty input is ignored (the entry is still marked `Notified`). Calling it with the same id again is a no-op.

claim (*group approvals only*)

Claims a group approval entry. This is the "first wins" step: once claimed, all other group members' entries for the same approval request are cancelled, and only the claimer can then approve or reject.

```
POST {baseUrl}/companies({companyId})/externalApprovalEntries(c4e7d1a2-0002-ef11-bf8d-6045bd028c9f)/Microsoft.NAV.claim
Content-Type: application/json

{
  "claimerEmail": "jane.smith@contoso.com"
}
```

`claimerEmail` is **required** — pass the email of the user who is claiming the entry. This is recorded in the audit log and written back to the BC approval entry as the claimer's identity.

```
Response `200 OK`:
```json
{
 "id": "c4e7d1a2-0002-ef11-bf8d-6045bd028c9f",
 "entryNo": 43,
 "status": "Claimed",
 "claimedByEmail": "jane.smith@contoso.com",

```

```
"claimedByDisplayName": "Jane Smith",
"claimedDateTime": "2026-05-04T10:41:02Z",
...
}
```

“ **Race condition:** If two group members call `claim` simultaneously, only one wins. The loser receives `400 Bad Request` with the error body: *"This entry was already claimed by jane.smith@contoso.com."* Your UI should catch this and refresh the entry for the losing caller.

## releaseClaim (group approvals only)

Releases a previously claimed entry, returning it to `Notified` status so another group member can claim it.

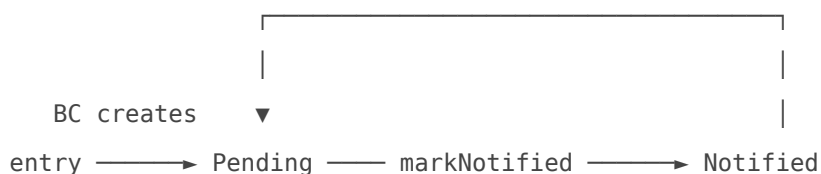
```
POST {baseUrl}/companies({companyId})/externalApprovalEntries(c4e7d1a2-0002-ef11-bf8d-6045bd028c9f)/Microsoft.NAV.releaseClaim
Content-Type: application/json

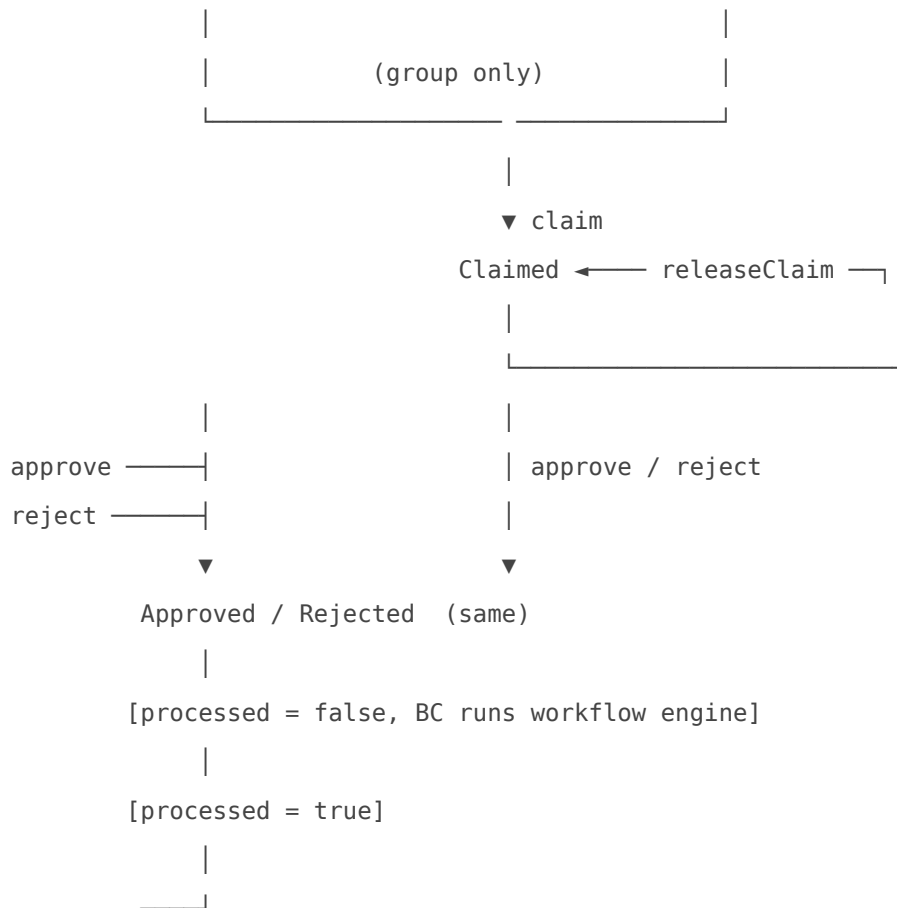
{}
```

Response `200 OK`:

```
{
 "id": "c4e7d1a2-0002-ef11-bf8d-6045bd028c9f",
 "entryNo": 43,
 "status": "Notified",
 "claimedByEmail": "",
 "claimedByDisplayName": "",
 ...
}
```

## Status State Machine





Status	Meaning	Actionable?
Pending	Created, not yet delivered to the approver	markNotified, approve, reject, claim
Notified	Your system confirmed the approver was notified	approve, reject, claim
Claimed	A group member has claimed this entry	approve, reject (claimer only), releaseClaim
Approved	Approver approved; BC may still be processing	—
Rejected	Approver rejected	—
Cancelled	Cancelled by BC (peer claimed/decided, workflow reassigned, or document re-opened)	—

“ **processed vs. status** : `status` is the approver's decision. `processed = true` means BC's workflow engine has successfully consumed the decision. An entry can be `status = Approved` with `processed = false` if the workflow engine call failed — check `errorMessage` for details. BC admins can retry from the External Approval Entries page.

# Group Approvals

When an approval stage targets an **External Approver Group**, BC creates one `externalApprovalEntry` for each group member. They share the same `workflowInstanceId` and `approvalEntryNo` but have different `id`, `approverEmail`, and `externalApproverCode` values.

`isGroupApproval = true` on all of them.

## Integration flow for group approvals

1. **Notify all members** — query entries by `extApproverGroupCode` + `status eq 'Pending'`. Show each member their own entry and store the entry's `id` for subsequent action calls.
2. **Member claims** — when a member taps "Claim", call `Microsoft.NAV.claim` on **their** entry's `id`. BC cancels the other members' entries automatically.
3. **Claimer decides** — show the approver their claimed entry. They call `approve` or `reject` on the same `id` they claimed.
4. **Poll for cancellations** — after a claim, entries belonging to other group members transition to `Cancelled`. Your UI should handle this gracefully (e.g. "This approval has been claimed by Jane Smith").

## Detecting peer cancellations

```
GET {baseUrl}/companies({companyId})/externalApprovalEntries
 ?$filter=approverEmail eq 'bob@contoso.com' and status eq 'Cancelled' and processed eq
false
 &$orderby=createdDateTime desc
 &$top=50
```

Entries in `Cancelled` state that have `processed = false` were cancelled because a peer claimed or decided first (not because the workflow was revoked by a BC user).

## Single-member groups

BC auto-claims single-member group entries at creation time — `status` is already `Claimed` when first fetched and `claimedByEmail` is pre-populated. Skip the claim step and go straight to `approve` / `reject`.

## Field Reference — externalApprovalEntry

JSON field	Type	Description
------------	------	-------------

id	GUID	<b>OData key.</b> SystemId — use this in all action URLs and single-record GETs.
entryNo	Integer	Internal BC sequence number. Useful for display, correlation, and reading related SharePoint items (BCEntryNo column). Not the OData key.
approvalEntryNo	Integer	Key of the linked BC standard Approval Entry.
workflowInstanceId	GUID	BC workflow instance. Shared by all entries (including related lines) in the same workflow run.
templateCode	String	AWF Workflow Template code.
stageCode	String	AWF Workflow Stage code.
stageDescription	String	Human-readable stage description. Show to the approver.
documentTableId	Integer	BC table ID of the source record (e.g. 38 for Purchase Header, 36 for Sales Header). Useful for routing or rendering table-specific UI in your portal.
documentNo	String	Document number being approved (e.g. P0-00123).
documentType	String	Document type enum caption (e.g. Order, Invoice, "" for custom tables).
externalApproverCode	String	Code of the External Approver record in BC.
extApproverGroupCode	String	Code of the External Approver Group, if this is a group approval. Empty for individual approvals.
approverEmail	String	Email of the approver assigned to this specific entry. Use for routing.
approverDisplayName	String	Display name of the assigned approver.
status	String	See <a href="#">Status State Machine</a> above.
isGroupApproval	Boolean	Whether this is part of a claim-based group approval.
claimedByEmail	String	Email of the group member who claimed the entry. Empty on individual approvals.
claimedByDisplayName	String	Display name of the claimer.
claimedDateTime	DateTime	When the entry was claimed (UTC).

<code>processed</code>	Boolean	Whether BC's workflow engine has successfully consumed this decision.
<code>processedDateTime</code>	DateTime	When BC processed the decision (UTC).
<code>errorMessage</code>	String	Non-empty if <code>processed = false</code> after an Approved/Rejected decision — the workflow engine call failed.
<code>description</code>	String	Contextual description (e.g. vendor name, customer name). Good for display in a card or list.
<code>recordDescription</code>	String	Full record identifier (e.g. "Table 38 (Purchase Header): Order, P0-00123"). Use for detailed display.
<code>amount</code>	Decimal	Approval amount.
<code>dueDate</code>	Date	Approval due date (ISO 8601 date, e.g. 2026-05-15 ).
<code>senderUserId</code>	String	BC User ID of the person who sent the document for approval.
<code>languageCode</code>	String	BC language code of the assigned approver (e.g. ENU , DEU ). Use to localise your notification.
<code>responseComment</code>	String	Comment entered by the approver. Populated after approve/reject.
<code>responseDateTime</code>	DateTime	When the approver responded (UTC).
<code>isRelatedApproval</code>	Boolean	True for line-level (related) entries linked to a header approval.
<code>createdDateTime</code>	DateTime	When BC created this entry (UTC).
<code>spListItemId</code>	String	SharePoint list item ID. Only populated when SP Approvals is enabled. Ignore for pure API integrations.
<code>spAttachmentsUrl</code>	String	SharePoint sharing link for document attachments. Only populated when SP Approvals is enabled and attachments were uploaded.

## Field Reference — externalApprover

JSON field	Type	Description
<code>id</code>	GUID	<b>OData key.</b> SystemId.
<code>code</code>	String	Internal BC approver code (Code[20]).

JSON field	Type	Description
displayName	String	Full name.
email	String	Email / UPN.
languageCode	String	BC language code for notifications.
entraObjectId	GUID	Entra ID object ID (populated for Entra-synced approvers).
blocked	Boolean	Blocked approvers cannot receive new entries.

## Field Reference — externalApproverMember

JSON field	Type	Description
id	GUID	<b>OData key.</b> SystemId.
groupCode	String	The External Approver Group code.
lineNo	Integer	Line number within the group (informational only).
externalApproverCode	String	Member's approver code.
displayName	String	Member's display name.
email	String	Member's email.

## Complete Example: Individual Approval Flow

The following sequence implements a minimal approval portal for individual approvers.

### Step 1 — Authenticate and get pending entries

```
GET {baseUrl}/companies({companyId})/externalApprovalEntries
 ?$filter=approverEmail eq 'john.doe@contoso.com'
 and status ne 'Cancelled'
 and processed eq false

&$select=id,entryNo,documentNo,description,stageDescription,amount,dueDate,status,isGroupApproval,createdDateTime
 &$orderby=createdDateTime asc
```

### Step 2 — Show entry details to the approver

Use `description`, `stageDescription`, `amount`, `dueDate`, and `senderUserId` to build an approval card. Store the entry's `id` GUID — you need it for action calls.

## Step 3 — Mark as notified

After showing the card to the approver:

```
POST {baseUrl}/companies({companyId})/externalApprovalEntries(b9f3a2c1-0001-ef11-bf8d-6045bd028c9f)/Microsoft.NAV.markNotified
Content-Type: application/json

{}
```

## Step 4 — Collect decision

Approver clicks Approve with a comment:

```
POST {baseUrl}/companies({companyId})/externalApprovalEntries(b9f3a2c1-0001-ef11-bf8d-6045bd028c9f)/Microsoft.NAV.approve
Content-Type: application/json

{
 "approverEmail": "john.doe@contoso.com",
 "comment": "Within budget allocation, approved."
}
```

---

# Complete Example: Group Approval Flow

## Step 1 — Notify all group members

Fetch entries for the group, one per member:

```
GET {baseUrl}/companies({companyId})/externalApprovalEntries
 ?$filter=extApproverGroupCode eq 'FINANCE-GROUP'
 and (status eq 'Pending' or status eq 'Notified')
 and processed eq false
```

Send each member a notification that includes their entry's `id`.

## Step 2 — Member claims

Jane opens the portal and taps "Claim" (using the `id` from her own entry):

```
POST {baseUrl}/companies({companyId})/externalApprovalEntries(c4e7d1a2-0002-ef11-bf8d-6045bd028c9f)/Microsoft.NAV.claim
Content-Type: application/json

{
 "claimerEmail": "jane.smith@contoso.com"
}
```

BC cancels all sibling entries (the other group members' entries for this same approval).

**\*\*Handle the race:\*\*** If the entry was already claimed by someone else, BC returns `400 Bad Request`:

```
```json
{
  "error": {
    "code": "Internal",
    "message": "This entry was already claimed by bob.smith@contoso.com. CorrelationId: ..."
  }
}
```

Refresh and show "This approval was claimed by Bob Smith" to Jane.

Step 3 — Claimer decides

Jane calls approve using the same `id`:

```
POST {baseUrl}/companies({companyId})/externalApprovalEntries(c4e7d1a2-0002-ef11-bf8d-6045bd028c9f)/Microsoft.NAV.approve
Content-Type: application/json

{
  "approverEmail": "jane.smith@contoso.com",
  "comment": "Reviewed line items, all within policy."
}
```

Error Handling

HTTP status codes

Code	When
200 OK	Action succeeded. Response body contains the updated entry.
400 Bad Request	Business logic error (entry already processed, already claimed by someone else, not in a claimable state). Read <code>error.message</code> for details.
401 Unauthorized	Missing or expired bearer token.
403 Forbidden	The API caller's BC user lacks the required permission set.
404 Not Found	Entry not found in this company, or wrong <code>id</code> GUID.

Idempotency

- `approve` and `reject` are guarded: calling them on an already-processed entry returns `400` with *"This external approval entry has already been processed..."*. Safe to retry only if the previous call returned a non-`200` response.
- `markNotified` is a no-op when status is not `Pending` — safe to call multiple times.
- `claim` is not idempotent by design: calling it twice from two different callers is the race condition you need to handle.

SharePoint/Teams race condition

When SharePoint integration is active, the entry may have been decided via Teams between your last poll and your `approve`/`reject`/`claim` call. The API syncs live SP state before processing (`SyncFromSPIfNeeded`), so a now-closed entry is rejected with `400` and one of the actual guard messages:

“*"This external approval entry has already been processed by Business Central and cannot be acted on again."*

"This external approval entry has been cancelled (a peer in the group decided first, or the workflow was reassigned). Refresh the list to see the current state."

This is not a retryable error — treat it like a `Cancelled` state and refresh the entry list for the user.

Entries that disappear

An entry's status can transition to `Cancelled` at any time due to external events:

- A BC user reassigns or cancels the underlying approval entry
- The document is re-opened (cancels the whole workflow)
- A peer in the same group claimed first (only for group entries)

Always check for `Cancelled` status before showing an entry to the approver, and handle `400` responses gracefully with a refresh.

Polling Recommendations

There is no webhook or push mechanism — your integration polls the API. Recommended intervals:

Use case	Interval
Notification delivery (fetch new <code>Pending</code> entries)	1-5 minutes
Approval portal refresh (logged-in user)	On demand (user refresh) + on page load
Group membership refresh	Query <code>externalApproverMembers</code> on each portal login

To detect newly created entries since the last poll, filter on `createdDateTime`:

```
GET {baseUrl}/companies({companyId})/externalApprovalEntries
   ?$filter=status eq 'Pending' and createdDateTime gt 2026-05-04T09:00:00Z
    &$orderby=createdDateTime asc
```

Record JSON

Each approval entry has a corresponding `externalApprovalEntryDocuments` endpoint that returns the full JSON representation of the source document record — the actual BC table row being approved (e.g. the Purchase Order, Sales Quote, or custom record).

This is a **separate endpoint** from `externalApprovalEntries`. It is only fetched when explicitly called, keeping list queries fast.

```
GET {baseUrl}/companies({companyId})/externalApprovalEntryDocuments(b9f3a2c1-0001-ef11-bf8d-6045bd028c9f)
```

The `id` is the same SystemId GUID as the corresponding `externalApprovalEntry`. Example response:

```
{
  "id": "b9f3a2c1-0001-ef11-bf8d-6045bd028c9f",
  "entryNo": 42,
  "documentNo": "P0-00123",
  "recordJson": "{\"No_\": \"P0-00123\", \"Document_Type\": \"Order\", \"Buy-
from_Vendor_No_\": \"V00010\", \"Buy-from_Vendor_Name\": \"Fabrikam
```

```
Inc.\",\"Amount\":15000.00,\"Amount_Including_VAT\":17850.00,\"Due_Date\":\"2026-05-15\", \"TableNo\":38,\"TableName\":\"Purchase Header\", \"TableCaption\":\"Purchase Header\", \"Company\":\"CRONUS International Ltd.\", ...}
}
```

“recordJson” is a JSON string (not a nested object) — parse it with `JSON.parse()` in your client.

What the JSON contains

The JSON object is produced by `DXP.Json.Helper.Rec2Json` and includes:

- **All table fields** — field captions (with spaces replaced by `_`) as keys, values serialized to their JSON-native types
- **Metadata fields** automatically added by the helper:
 - `TableNo` — numeric table ID
 - `TableName` — internal table name
 - `TableCaption` — localized table caption
 - `Company` — company name
 - `RecordId` — BC RecordId string
 - `SysLink` — SystemId + TableNo composite

Performance note

`recordJson` is computed on every record read — it navigates to the source document and serializes all its fields. Since it lives on a dedicated endpoint, it is only fetched when you explicitly call `externalApprovalEntryDocuments`. Use the entries endpoint for list queries and only call the documents endpoint when you need the full record:

```
# Fast list – no document lookup
GET {baseUrl}/companies({companyId})/externalApprovalEntries
    ?$filter=approverEmail eq 'john@contoso.com' and status eq 'Pending'

# On-demand – fetch the full record only for the entry the approver opened
GET {baseUrl}/companies({companyId})/externalApprovalEntryDocuments({id})
```

Fallback

If the source document was deleted or the Approval Entry link is missing, `recordJson` returns `"{}"` (an empty JSON object string) rather than an error.

Record Attachments

Each approval entry has a corresponding `externalApprovalEntryAttachments` endpoint that returns all standard BC document attachments for the source record as a JSON array. Each element includes the file metadata and the full file content as a Base64 string, ready to render or download client-side.

This is a **separate endpoint** from `externalApprovalEntries`. It is only fetched when explicitly called.

```
GET {baseUrl}/companies({companyId})/externalApprovalEntryAttachments(b9f3a2c1-0001-ef11-bf8d-6045bd028c9f)
```

Example response:

```
{
  "id": "b9f3a2c1-0001-ef11-bf8d-6045bd028c9f",
  "entryNo": 42,
  "documentNo": "P0-00123",
  "recordAttachmentsJson": "[{\"fileName\":\"P0-00123 Specification\",\"fileExtension\":\"pdf\",\"attachedDate\":\"2026-05-03T14:22:00Z\",\"attachedBy\":\"Alice Johnson\",\"lineNo\":0,\"contentBase64\":\"JVBERi0xLjQK...\"},{\"fileName\":\"Vendor Quote\",\"fileExtension\":\"xlsx\",\"attachedDate\":\"2026-05-03T14:23:00Z\",\"attachedBy\":\"Alice Johnson\",\"lineNo\":0,\"contentBase64\":\"UESDBBQA...\"}]"
}
```

“ `recordAttachmentsJson` is a JSON string — parse it with `JSON.parse()` .

Attachment object fields

Field	Type	Description
<code>fileName</code>	String	File name without extension.
<code>fileExtension</code>	String	File extension without dot (e.g. <code>pdf</code> , <code>xlsx</code> , <code>png</code>).
<code>attachedDate</code>	DateTime	When the file was attached (UTC).
<code>attachedBy</code>	String	BC user ID of the person who attached the file.

Field	Type	Description
lineNo	Integer	0 for document-level (header) attachments; non-zero for line-level attachments.
contentBase64	String	Full file content, Base64-encoded. Empty string if the file has no content. Decode with <code>atob()</code> in a browser or your platform's Base64 decoder.

Client-side decode example (JavaScript)

```
const attachments = JSON.parse(entry.recordAttachmentsJson);
attachments.forEach(a => {
  const bytes = Uint8Array.from(atob(a.contentBase64), c => c.charCodeAt(0));
  const blob = new Blob([bytes], { type: mimeTypeFor(a.fileExtension) });
  const url = URL.createObjectURL(blob);
  // render download link or open inline
});
```

Performance note

`recordAttachmentsJson` reads and Base64-encodes every attachment for the source document on every record read. Since it lives on a dedicated endpoint, it is only fetched when you explicitly call `externalApprovalEntryAttachments`:

```
# Fast list – no attachment loading
GET {baseUrl}/companies({companyId})/externalApprovalEntries
  ?$filter=approverEmail eq 'john@contoso.com' and status eq 'Pending'

# On-demand – fetch attachments only when the approver opens an entry
GET {baseUrl}/companies({companyId})/externalApprovalEntryAttachments({id})
```

Fallback

Returns `[]` (empty JSON array string) if the source document has no attachments, or if the document was deleted.

Related Documentation

Document	Purpose
SETUP-GUIDE.md	Admin setup guide for the SharePoint / Teams integration

Document	Purpose
SHAREPOINT-CONTRACT.md	Contract for integrations that read/write the SharePoint list directly
POWERAUTOMATE-DEV-GUIDE.md	Internal guide for maintaining the Power Automate reference flow

Revision #2
Created 2026-06-24 12:18:45 UTC by Bernd Feddersen
Updated 2026-06-24 12:18:49 UTC by Bernd Feddersen