外部公開API機能利用マニュアル

OS	
最終更新日時	2026年11月5日 10:53

外部公開API機能利用マニュアル

第7版

本書は、セキュアSAMBAの外部公開APIの利用方法（利用開始・APIキー管理）と、各エンドポイントの仕様を説明します。

外部公開APIの利用開始

外部公開APIは、貴社がシステム連携目的で利用するAPIです。アクセスには **APIキー**（`kbs_` プレフィックスの文字列）が必要です。

利用申請と有効化

外部公開APIのご利用にはお申し込みが必要です。kubellストレージへご利用をお申し込みください。

お申し込み後、kubell側で貴社の外部API利用が「利用可能」に設定されると、機能が有効化されます。

- 利用可能になると、貴社の管理画面に **「APIキー管理」メニュー** が表示され、**オーナー** がAPIキーを発行・管理できるようになります。
- ご利用を停止（「利用不可」に変更）した場合、**発行済みのAPIキーはすべて無効化** されます。再開には改めて「利用可能」への設定が必要です。

APIキーの管理（オーナー向け）

APIキーの発行・停止は、貴社の **オーナー** が自社の管理画面の「APIキー管理」メニューから行います。

重要事項

- APIキーは **発行時の1回限り** 平文で表示されます。発行後は再表示できません。安全な方法で速やかに保管し、利用するシステムに設定してください。
- APIキーによるリクエストは、**キーに紐づけたユーザーの権限** で処理されます。アクセスさせたいフォルダに適切な権限（閲覧・編集など）を持つユーザーを紐づけてください。
- **1ユーザーにつき有効なAPIキーは1つまで** です。既に有効なキーがあるユーザーには追加発行できません。新たに発行するには、既存キーを停止してください。停止済み・有効期限切れのキーのみの場合は新規発行できます。
- 一度発行したAPIキーの内容（紐付けユーザー・有効期限など）は **変更できません**。変更したい場合は停止して新規発行してください。
- 紐付けユーザーのアカウントが削除されると、そのユーザーに紐づくAPIキーは **自動的に無効化** されます。
- 発行・停止の操作は **操作ログに記録** されます（発行時は有効期限も記録）。なお、有効期限切れによる自然失効は操作ログには記録されません。

APIキーの発行

「APIキー管理」メニューから新しいAPIキーを発行します。発行時に以下を指定します。

項目	必須	説明
紐づけるユーザー	✓	キーに紐づける自社の顧客ユーザー。リクエストはこのユーザーの権限で処理されます
キー名	✓	キーの識別名。`本番用`、`テスト用` など用途がわかる名称を推奨
有効期限	-	省略した場合は無期限。セキュリティ上、可能な限り設定を推奨

発行直後に表示される平文のAPIキー（`kbs_` で始まる文字列）を必ず控えてください。この画面を閉じると再表示できません。

APIキーの一覧確認

発行したAPIキーの一覧を確認できます。各キーの名前、識別用プレフィックス、紐付けユーザー、有効期限（`expires_at`）、失効日時（`revoked_at`）、最終利用日時、作成日時などが表示されます。平文のAPIキーは一覧には表示されません。

一覧には、現在有効なキーだけでなく、**停止（失効）済み・有効期限切れのキーも履歴として表示** されます。各キーが有効か無効かは、`revoked_at`（失効日時が入っていれば停止済み）と `expires_at`（過去日時であれば期限切れ）で判別できます。

APIキーの停止

不要になったキーや漏洩が疑われるキーは、一覧から停止（無効化）できます。停止したキーは即時利用不可になります。個別のAPIキーの発行・停止は貴社オーナーご自身で行ってください。

> **緊急時の対応について**
> kubellストレージ側で対応できるのは外部APIの利用そのものの **ON/OFF（利用可能／利用不可）のみ** であり、個別のAPIキーの管理には関与できません。
> 緊急時にお客様からご要望があれば、kubell側で外部APIの利用を **OFF（利用不可）** にすることは可能です。ただしこの場合、**発行済みのすべてのAPIキーが無効化** され、利用を再開する際には **すべてのAPIキーを改めて発行し直す必要** がある点にご留意ください。

運用ガイドライン

- **紐付けユーザーの権限を事前に確認する。** APIキーは紐づいたユーザーの権限で動作します。アクセスさせたいフォルダに対して適切な権限が付与されているか確認してください。
- **有効期限を設定する。** セキュリティ上、無期限キーは極力避け、利用期間が明確な場合は有効期限を設定してください。
- **キー名は用途がわかるよう命名する。** 管理しやすい名前を付けてください。
- **漏洩時は速やかに停止する。** 該当キーを停止し、必要に応じて新しいキーを再発行してください。

外部公開API リファレンス

ここからは、APIキーを用いて呼び出す各エンドポイントの仕様を説明します。

はじめに

ベースURL

```
https://external-api.securesamba-cloud.com/api
```

認証

すべてのAPIリクエストには、発行されたAPIキーをBearerトークンとしてリクエストヘッダーに含めてください。

```
Authorization: Bearer kbs_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

APIキーが無効・期限切れ・失効済みの場合は `401 Unauthorized` が返ります。

> **【ご注意：Web画面のアクセス制限は適用されません】** 外部公開APIへのリクエストには、管理画面で設定する **IPアドレス制限・端末制限・2要素認証は適用されません**。有効なAPIキーがあれば、許可していないIPアドレスや端末からも利用できます。APIキーは第三者に知られないよう厳重に管理し、不要になったキーや漏えいのおそれがあるキーは速やかに停止してください（「APIキーの停止」を参照）。

APIキーは、貴社のオーナーが自社の管理画面「APIキー管理」メニューから発行します。発行手順や注意事項は本書の「APIキーの管理（オーナー向け）」を参照してください。なお、外部公開APIのご利用開始にはお申し込みが必要です（「外部公開APIの利用開始」を参照）。

レート制限

APIはAPIキーごとに1分あたりのリクエスト数が制限されています。制限を超えると `429 Too Many Requests` が返ります。上限値は法人により異なる場合があります。

共通エラーレスポンス

ステータス	説明
401	認証エラー（APIキーが無効・欠落）
403	権限エラー（操作に必要な権限がない、保護されているなど）
404	リソースが見つからない（または権限外のリソースへのアクセス）
422	バリデーションエラー（リクエストパラメータが不正）
423	契約容量・ファイル数・フォルダの容量上限などの上限に達しているため操作できない
429	レート制限超過
500	サーバー内部エラー

401 エラーレスポンス例

```
{
  "error": {
    "code": "UNAUTHORIZED",
    "message": "APIキーが必要です。Authorization: Bearer <api_key> ヘッダーを指定してください。"
  }
}
```

422 エラーレスポンス例

```
{
  "message": "The name field is required.",
  "errors": {
    "name": ["The name field is required."]
  }
}
```

フォルダの容量上限

共有フォルダ直下のフォルダに容量上限が設定されている法人では、使用量が上限に達したフォルダの配下（フォルダ自身を含む）で次の操作を行うと、`423` とエラーコード `ERROR_CODE_FOLDER_CAPACITY_LIMIT_REACHED` が返ります。

- フォルダ作成（一括作成を含む）・ファイルアップロード（シンプル・チャンクの init）
- リネーム・ノードコピー・プレビューURL取得
- ノード移動（移動先が容量上限に達したフォルダの配下の場合が対象です。外から中への移動に加えて、同じフォルダの中での移動もできません。フォルダの中から外への移動はできます）
- ノードのタグ更新
- 権限の一覧取得・付与・変更・削除

ノードの取得・検索・ダウンロード・削除（ゴミ箱へ移動）・ノードのタグ取得・タグの作成・名前の変更・削除・一覧取得・お気に入りの操作はできます。チャンクアップロードの chunk・finalize では判定しません（init で判定します）。

```
{
  "errors": {
    "message": "フォルダ「営業部」の容量上限に達しています。",
    "code": "ERROR_CODE_FOLDER_CAPACITY_LIMIT_REACHED"
  }
}
```

> APIキーの利用者が容量上限に達したフォルダを閲覧できない場合は、フォルダ名を含まない「このフォルダの容量上限に達しています。」が返ります。

`errors` 形式のエラーレスポンス

権限・ユーザー・グループの各エンドポイントは、エラーの内容を `errors`（**複数形**）オブジェクトで返します。`message` に日本語の説明文、`code` にエラーコードが入ります。

```
{
  "errors": {
    "message": "「営業資料」が含まれる親フォルダの権限設定により、これより低い権限を付与することはできません。親フォルダと同等、またはそれ以上の権限を指定してください。",
    "code": "ERROR_CODE_PERMISSION_CHANGE_ANCESTORS"
  }
}
```

> リクエストパラメータの形式そのものが不正な場合（`permission_level_code` に許可されていない値を指定した、など）は、この形式ではなく上記の **422 エラーレスポンス例** の形式で返ります。

エラーコード一覧（権限・ユーザー・グループ）

権限・ユーザー・グループの各エンドポイントが返すエラーコードは以下のとおりです。

コード	意味	ステータス
`ERROR_CODE_COMPANY_STORAGE_LIMIT_REACHED`	契約容量の上限に達しているため操作できない	423
`ERROR_CODE_COMPANY_NODE_COUNT_LIMIT_REACHED`	ファイル・フォルダ数の上限に達しているため操作できない	423
`ERROR_CODE_FOLDER_CAPACITY_LIMIT_REACHED`	対象ノードが容量上限に達したフォルダの配下にあるため操作できない	423
`ERROR_CODE_NODE_NOT_FOUND`	指定したノードが存在しない（他社のノード・削除済みのノードを含む）	404
`ERROR_CODE_NODE_IN_TRASH`	指定したノードがゴミ箱内にある（ゴミ箱そのものを指定した場合を含む）	404
`ERROR_CODE_RESOURCE_NOT_FOUND`	指定したユーザー・グループが存在しない（他社・削除済みを含む）	404
`ERROR_CODE_PERMISSION_DENIED`	対象ノードの権限を変更できる権限がない、またはファイル・フォルダの共有が制限されている	403
`ERROR_CODE_NODE_PROTECTED`	対象ノードまたはその配下に、他のユーザーが保護したノードがある	403
`ERROR_CODE_UPLOADING`	対象のファイルがアップロード中	400
`ERROR_CODE_NODE_MAX_DEPTH_EXCEEDED`	対象ノードが階層数の上限を超えた位置にある	400
`ERROR_CODE_USER_SUSPENDED`	対象ユーザーが停止中のため権限を付与・変更できない	400
`ERROR_CODE_PERMISSION_CHANGE_SAME`	上位フォルダから継承している権限と同じ権限には変更できない	400
`ERROR_CODE_PERMISSION_CHANGE_ANCESTORS`	上位フォルダから継承している権限より低い権限は指定できない／継承している権限はこの階層では削除できない	400
`ERROR_CODE_PERMISSION_CAN_NOT_CHANGE`	所属グループ経由で適用されている権限と同じか、それより低い権限には変更できない	400
`ERROR_CODE_PERMISSION_CAN_NOT_GRANT`	所属グループ経由で適用されている権限と同じか、それより低い権限は付与できない	400
`ERROR_CODE_PERMISSION_ADMIN_MUST_REMAIN`	操作を行うと、対象ノードの管理権限を持つユーザーが1人もいなくなる	400
`ERROR_CODE_PERMISSION_STILL_REMAIN`	削除前の確認。削除しても上位フォルダから継承している権限が残る	400
`ERROR_CODE_PERMISSION_GROUP_USER_DELETE_SAME`	削除前の確認。削除しても所属グループ経由の権限が残る	400

コード	意味	ステータス
`ERROR_CODE_PERMISSION_DELETE`	削除前の確認。対象ノードに直接設定された権限を削除してよいか	400
`ERROR_CODE_PERMISSION_CAN_NOT_DELETE`	対象ノードに削除できる権限が直接設定されていない	400
`ERROR_CODE_GROUP_PROTECTED`	保護グループ「すべてのユーザー」は更新・削除・所属ユーザーの追加・削除ができない	400
`ERROR_CODE_GROUP_LIMIT_REACHED`	グループの作成数が上限（100件。保護グループを含む）に達している	400
`EXTERNAL_SHARE_LINK_APPROVAL_SETTING_DELETION_DENIED`	「リンクの発行申請」が有効で、削除しようとしたグループが唯一の「申請が必須になるグループ・ユーザー」に設定されている	422

> `EXTERNAL_SHARE_LINK_APPROVAL_SETTING_DELETION_DENIED` はステータスが 422 ですが、リクエストパラメータの形式エラー（422 エラーレスポンス例）とは異なり、上記の `errors` **形式** で返ります。

ノード（ファイル・フォルダ）

NodeResource（ノードオブジェクト）

各APIのレスポンスに含まれるノードオブジェクトの構造です。

フィールド	型	説明
`reference_id`	string	ノードの識別ID（UUID形式）。各APIのパスパラメータに使用します
`name`	string	ノード名
`full_path`	string	ノードのフルパス。先頭スラッシュから始まり、`/` 区切りで対象ノード自身の名前までを含みます（例: `/共有フォルダ/営業部/資料.pdf`）
`node_type`	string	種別。`folder`（フォルダ）または `file`（ファイル）、`base`（ルートノード）
`placement`	string	配置場所。`shared_folder`（共有フォルダ配下）、`my_folder`（マイフォルダ配下）、`trash`（ゴミ箱配下）
`mime_type`	`string | null`	MIMEタイプ（ファイルのみ。フォルダは `null`）
`size`	`integer | null`	ファイルサイズ（バイト）。フォルダは `null`
`is_protected`	boolean	保護設定の有無
`created_at`	string	作成日時（ISO 8601形式）
`updated_at`	string	更新日時（ISO 8601形式）
`created_by`	`object | null`	作成者情報。`id`（integer）と `name`（string）を含む

- `full_path` について、APIキーの利用者が閲覧権限を持たない中間フォルダは、パスから除かれます。そのため実際の格納階層と一致しない場合があります。

フルパスの長さの上限

ファイル・フォルダのフルパスの長さには、Windows のパスの長さの上限に合わせて **259文字** の上限があります。上限を超える操作は `400` `ERROR_CODE_FULL_PATH` になり、行われません。

- 先頭にドライブ文字 `V:` を付けたフルパスの文字数で判定します（例: `V:/共有フォルダ/営業部/資料.pdf`）。`full_path` と同じ形式（`/` 始まり）で数えると、**257文字** までになります。
- 半角・全角を問わず、1文字を1文字として数えます。
- 実際の格納階層で判定します。`full_path` には閲覧権限のない中間フォルダが含まれないため、`full_path` が上限より短くてもこのエラーになることがあります。
- フォルダのリネーム・移動・コピーでは、配下のファイル・フォルダのフルパスも判定します。配下のどれか1件でも上限を超える場合は、操作全体が行われません。

この上限は、次のエンドポイントで判定します。

- フォルダ作成
- フォルダー括作成（並列）

- ファイルアップロード（シンプル・15MBまで）
- チャンクアップロードの init
- リネーム
- ノード移動
- ノードコピー

エラーレスポンス例

```
{
  "errors": {
    "message": "パスの最大数を超えています。",
    "code": "ERROR_CODE_FULL_PATH"
  }
}
```

ベースノード取得

ルートノード（共有フォルダ・マイフォルダ）の情報を取得します。

ファイル・フォルダの一覧取得を行う際の起点となる `reference_id` を確認するために使用します。

マイフォルダの利用が制限されているユーザーの場合、レスポンスには共有フォルダのみ含まれます。

```
GET /v1/base_nodes
```

レスポンス（200 OK）

```
{
  "data": [
    {
      "reference_id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
      "name": "共有フォルダ",
      "full_path": "/共有フォルダ",
      "node_type": "base",
      "mime_type": null,
      "size": null,
      "is_protected": false,
      "created_at": "2025-01-01T00:00:00+09:00",
      "updated_at": "2025-01-01T00:00:00+09:00",
      "created_by": null
    },
    {
      "reference_id": "yyyyyyyy-yyyy-yyyy-yyyy-yyyyyyyyyyyy",
      "name": "マイフォルダ",
      "full_path": "/マイフォルダ",
      "node_type": "base",
      "mime_type": null,
      "size": null,
      "is_protected": false,
      "created_at": "2025-01-01T00:00:00+09:00",
      "updated_at": "2025-01-01T00:00:00+09:00",
      "created_by": {
        "id": 1,
        "name": "山田 太郎"
      }
    }
```

```
  ]
}
```

フォルダ内一覧取得

指定したフォルダの直下にあるファイル・フォルダの一覧を取得します。

```
GET /v1/nodes/{referenceId}/children
```

パスパラメータ

パラメータ	型	必須	説明
`referenceId`	string (UUID)	✓	一覧を取得するフォルダの `reference_id`

レスポンス（200 OK）

```
{
  "data": [
    {
      "reference_id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
      "name": "サブフォルダ",
      "full_path": "/共有フォルダ/営業部/サブフォルダ",
      "node_type": "folder",
      "mime_type": null,
      "size": null,
      "is_protected": false,
      "created_at": "2025-01-01T00:00:00+09:00",
      "updated_at": "2025-01-01T00:00:00+09:00",
      "created_by": {
        "id": 1,
        "name": "山田 太郎"
      }
    }
  ]
}
```

エラーケース

ステータス	コード	説明
404	`ERROR_CODE_NODE_NOT_FOUND`	指定したノードが存在しない、または閲覧権限がない（他社のノードを含む）
404	`ERROR_CODE_NODE_IN_TRASH`	指定したノードがゴミ箱内にある
403	`ERROR_CODE_PERMISSION_DENIED`	「マイフォルダを非表示にする」制限が設定されているユーザーが、マイフォルダまたはその配下のフォルダを指定した

ノード詳細取得

指定したノード（ファイルまたはフォルダ）の詳細情報を取得します。

```
GET /v1/nodes/{referenceId}
```

パスパラメータ

パラメータ	型	必須	説明
`referenceId`	string (UUID)	✓	取得するノードの `reference_id`

レスポンス（200 OK）

```
{
  "data": {
    "reference_id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "name": "報告書.pdf",
    "full_path": "/共有フォルダ/営業部/報告書.pdf",
    "node_type": "file",
    "mime_type": "application/pdf",
    "size": 204800,
    "is_protected": false,
    "created_at": "2025-01-01T00:00:00+09:00",
    "updated_at": "2025-04-01T12:00:00+09:00",
    "created_by": {
      "id": 1,
      "name": "山田 太郎"
    }
  }
}
```


エラーケース

ステータス	コード	説明
404	`ERROR_CODE_NODE_NOT_FOUND`	指定したノードが存在しない、または閲覧権限がない（他社のノードを含む）
404	`ERROR_CODE_NODE_IN_TRASH`	指定したノードがゴミ箱内にある
403	`ERROR_CODE_PERMISSION_DENIED`	「マイフォルダを非表示にする」制限が設定されているユーザーが、マイフォルダまたはその配下のノードを指定した
400	`ERROR_CODE_UPLOADING`	指定したファイルがアップロード中
423	`ERROR_CODE_PROCESSING_VIRUS_SCAN`	指定したファイルがウイルスチェック中

> チャンクアップロードで init を呼び出してから finalize が完了するまでのファイルは、アップロード中として扱われます。

ノード検索

名前・日時・タグ名の条件で、ファイル・フォルダを検索します。3つの条件は**任意に組み合わせて指定でき**、複数指定した場合は**指定したすべての条件を満たす**ノードだけを返します（AND検索）。

操作者（APIキーに紐づくユーザー）が閲覧権限を持つノードのみが検索結果に含まれます。アクセスできないノードは結果に表示されません。ゴミ箱内のノードは対象外です。

```
GET /v1/nodes/search
```

クエリパラメータ

検索条件は以下の3グループです。**1つ以上**を指定してください（組み合わせは自由です）。いずれも指定しない場合は `422` を返します。

グループ	パラメータ	型	説明
名前	`name`	string	ノード名（フォルダ名・ファイル名）で検索（最大20文字）
日時	`time_type`	string	検索対象の日時。`created_at`（登録日時）または `last_updated_at`（最終更新日時）
日時	`from`	string (ISO 8601)	範囲の開始日時。タイムゾーンを明示すること（下記参照）
日時	`to`	string (ISO 8601)	範囲の終了日時。タイムゾーンを明示すること（下記参照）
タグ名	`tag_name`	string	タグ名（完全一致）。指定したタグが付与されているノードを返す（最大20文字。法人により異なる場合があります）

- **日時検索**: `time_type` は必須で、`from` / `to` の少なくとも一方を指定します（両方指定で範囲、片方のみも可）。どちらも指定しない場合は `422` を返します。`to` は `from` 以降である必要があります。また `from` は現在以前、`to` は UTC における当日の終わり（`23:59:59Z`）以前である必要があります。
- **日時のタイムゾーン指定**: 日時は ISO 8601 形式で、**タイムゾーンを必ず明示**します。指定したタイムゾーンはサーバー側で正しく解釈されます（保存はUTCで行われますが、指定タイムゾーンとの差分は自動変換されます）。
  - **UTC で指定する場合**: 末尾に `Z` を付けます（例: `2026-06-01T00:00:00Z`）
  - **日本時間（JST）で指定する場合**: オフセット `+09:00` を付けます（例: `2026-06-01T09:00:00+09:00` は UTC の `2026-06-01T00:00:00Z` と同じ時刻）
  - タイムゾーン（`Z` またはオフセット）のない日時（例: `2026-06-01T00:00:00`）は受け付けません。
- **タグ名検索**: 自社に**完全一致するタグ名が存在しない場合は** `404` を返します（タグ一覧取得で確認できる名前を指定してください）。タグ名検索は「タグが割り当てられているノードを探す」機能です。他の条件と組み合わせた場合も、タグ名が存在しなければ `404` を返します（タグは存在し、他の条件との組み合わせで該当が0件のときは `200` と空配列を返します）。
- **条件の組み合わせ**: 日時のパラメータ（`time_type` / `from` / `to`）を1つでも指定した場合は、他の条件と併用していても日時検索の条件（`time_type` 必須など）が適用されます。

検索の範囲は、次のパラメータで指定したフォルダの配下に絞り込めます（任意）。`root` は検索条件ではないため、`root` だけを指定した場合（名前・日時・タグ名のいずれも指定しない場合）は `422` を返します。

パラメータ	型	説明
`root`	string (UUID)	検索起点にするフォルダの `reference_id`。指定したフォルダの配下だけを検索する（指定できるのは1つだけ）

- **検索の範囲**: 指定したフォルダの配下の**全階層**（直下だけでなく、その中のフォルダの配下も含む）から、指定した条件をすべて満たすノードを返します。`root` に指定したフォルダ自身は、条件に合っていても結果に含まれません。配下かどうかはフォルダの階層上の位置で判定し、ショートカットのリンク先はたどりません。
- **指定できるノード**: フォルダと、ベースノード取得で得られる共有フォルダ・マイフォルダの `reference_id` です。ファイル・ショートカットを指定した場合は `422` を返します。また、`root` に指定するノードには操作者の閲覧権限が必要です。「マイフォルダを非表示にする」制限が設定されているユーザーは、マイフォルダとその配下のノードを指定できません（`403`。ファイル・ショートカットを指定した場合も `422` ではなく `403` を返します）。

リクエスト例

```
GET /v1/nodes/search?name=見積書
# UTC で指定
GET /v1/nodes/search?time_type=created_at&from=2026-06-01T00:00:00Z&to=2026-06-30T23:59:59Z
# 日本時間（JST）で指定
GET /v1/nodes/search?time_type=created_at&from=2026-06-01T00:00:00%2B09:00&to=2026-06-30T23:59:59%2B09:00
GET /v1/nodes/search?tag_name=重要
# 名前とタグ名の組み合わせ（両方を満たすノードのみ）
GET /v1/nodes/search?name=見積書&tag_name=重要
# 名前・日時・タグ名の3条件の組み合わせ
GET /v1/nodes/search?name=見積書&time_type=created_at&from=2026-06-01T00:00:00Z&to=2026-06-30T23:59:59Z&tag_name=重要
# 検索起点（root）に指定したフォルダの配下だけを、タグ名で検索
GET /v1/nodes/search?tag_name=重要&root=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

> URL のクエリ文字列ではオフセットの `+` を `%2B` にURLエンコードしてください（`+` はスペースとして解釈されるため）。

レスポンス（200 OK）

NodeResource の配列を返します。

```
{
  "data": [
    {
      "reference_id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
      "name": "見積書.pdf",
      "full_path": "/共有フォルダ/営業部/見積書.pdf",
```

```
      "node_type": "file",
      ...
    }
  ]
}
```

- 名前・日時検索（タグ名を指定しない場合）は、アクセス可能なノードのうち**最大1000件**（最終更新日時の新しいものから）を返します。
- **タグ名のみ**を指定した場合は、該当する全ノードを返します（件数の上限はありません）。並び順はノードが登録された順（古いものから）で、最終更新日時の新しい順ではありません。
- **タグ名を名前・日時と組み合わせた場合**は、条件を満たすノードのうち**最大1000件**（最終更新日時の新しいものから）を返します。1000件を超える該当がある場合、1000件目より後のノードは返りません。
- **検索起点（`root`）を指定した場合**は、条件の組み合わせにかかわらず（タグ名と `root` だけを指定した場合も含みます）、条件を満たすノードのうち**最大1000件**（最終更新日時の新しいものから）を返します。1000件を超える該当がある場合、1000件目より後のノードは返りません。

エラーケース

ステータス	コード	説明
422	（バリデーション）	条件未指定（`root` だけの指定を含む）／`name`・`tag_name` の文字数超過（各20文字まで。`tag_name` は法人により異なる場合があります）／`time_type` 未指定／`from`・`to` がどちらも未指定／日時形式不正（タイムゾーン未指定含む）／未来日時の指定（`from` が現在より後、または `to` が UTC における当日の終わりより後）／`to` が `from` より前／`root` の形式不正（UUID 形式でない）／`root` にフォルダ以外（ファイル・ショートカット）を指定した（`errors.root` に「検索起点がフォルダではありません。」）
404	`ERROR_CODE_NODE_NOT_FOUND`	`root` に指定したノードが存在しない（他社のノード、閲覧権限の無いノードを含む）
404	`ERROR_CODE_NODE_IN_TRASH`	`root` に指定したノードがゴミ箱内にある（ゴミ箱そのものの指定を含む）
403	`ERROR_CODE_PERMISSION_DENIED`	「マイフォルダを非表示にする」制限が設定されているユーザーが、`root` にマイフォルダまたはその配下のノードを指定した
404	`ERROR_CODE_RESOURCE_NOT_FOUND`	`tag_name` に完全一致するタグが存在しない

フォルダ作成

指定したフォルダの直下に新しいフォルダを作成します。

`name` に `/` 区切りで複数階層のフォルダ名を指定すると、その階層分のフォルダを親から子の順にまとめて新規作成できます（例: `2025年度資料/契約書/締結済` を指定すると 3 階層のフォルダを一度に作成）。

```
POST /v1/nodes/{parentReferenceId}/folders
Content-Type: application/json
```

パスパラメータ

パラメータ	型	必須	説明
`parentReferenceId`	string (UUID)	✓	作成先の親フォルダの `reference_id`

リクエストボディ

フィールド	型	必須	説明
`name`	string	✓	フォルダ名。`/` で区切ると複数階層をまとめて作成します。各階層のフォルダ名は最大191文字

- 各階層のフォルダ名（`/` で区切った各セグメント）には、フォルダ名として使用できない文字（`\ / : * ? " < > |` など）を含められません。
- 先頭・末尾の `/`、および `//` のように空の階層を含むパスは指定できません（422 エラー）。
- 既存のフォルダは再利用されず、指定したパスはすべて新規に作成されます。そのため、最初の階層（指定した親フォルダの直下）に同名フォルダが既に存在する場合は作成全体が失敗します（409 エラー）。

リクエスト例

```
{
  "name": "2025年度資料/契約書/締結済"
}
```

レスポンス（201 Created）

作成されたフォルダの NodeResource を**配列**で返します。配列は親から子の順（指定したパスの順）に並びます。`/` を含まない単一フォルダ作成の場合も、要素 1 件の配列を返します。

```
{
  "data": [
    {
      "reference_id": "yyyyyyyy-yyyy-yyyy-yyyy-yyyyyyyyyyyy",
      "name": "2025年度資料",
      "full_path": "/共有フォルダ/2025年度資料",
      "node_type": "folder",
      ...
    },
    {
      "reference_id": "zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzzzzzz",
      "name": "契約書",
      "full_path": "/共有フォルダ/2025年度資料/契約書",
      "node_type": "folder",
      ...
    },
    {
      "reference_id": "wwwwwwww-wwww-wwww-wwww-wwwwwwwwwwww",
      "name": "締結済",
      "full_path": "/共有フォルダ/2025年度資料/契約書/締結済",
      "node_type": "folder",
      ...
    }
  ]
}
```

> **【ご注意：レスポンス形式の変更】** 本エンドポイントは、複数階層の一括作成に対応するため、`data` を単一オブジェクトから**配列**に変更しました。単一フォルダ作成時も要素 1 件の配列で返却されます。既存の連携で `data` を単一オブジェクトとして参照している場合は、配列の先頭要素（`data[0]`）を参照するよう修正してください。

エラーケース

ステータス	コード	説明
422	（バリデーション）	`name` が未指定、または空の階層を含むなどフォルダ名が不正
409	`CONFLICT`	最初の階層に同名のフォルダが同じ場所に存在する
400	`ERROR_CODE_NODE_MAX_DEPTH_EXCEEDED`	作成によりフォルダ階層が上限（10階層）を超える
400	`ERROR_CODE_FULL_PATH`	作成されるフォルダのフルパスが上限（259文字）を超える（フルパスの長さの上限）
403	`ERROR_CODE_BULK_COPY_LOCKED`	作成先が「まとめてアップロード」の実行中のフォルダ、またはその配下のため操作できない
423	`ERROR_CODE_FOLDER_CAPACITY_LIMIT_REACHED`	対象が容量上限に達したフォルダの配下にある（フォルダの容量上限）

```
{
  "error": {
    "code": "CONFLICT",
```

```
    "message": "同名のアイテムがすでに存在します。"
  }
}
```

> 複数階層をまとめて作成する場合、フォルダ階層は最大 10 階層までです。途中の階層で上限を超えると、その時点で作成全体が失敗し、それまでに作成された階層も含めてすべて作成されません（ロールバックされます）。例えば 8 階層目のフォルダを親に指定して 5 階層分を一括作成しようとすると、11 階層目にあたる 3 つ目のフォルダ作成で上限を超えるため、リクエスト全体が失敗します。
>
> フルパスの長さの上限（259文字）も、作成するすべての階層で判定します。途中の階層のフォルダでフルパスが上限を超える場合も、リクエスト全体が失敗し、どの階層も作成されません。

フォルダー括作成（並列）

指定したフォルダの直下に、複数のフォルダを 1 リクエストでまとめて作成します。

前述のフォルダ作成が「親から子へ」の**階層（直列）**をまとめて作成するのに対し、こちらは同じ親の直下に**並ぶフォルダ（並列）**をまとめて作成します。作成されるのは指定した親の 1 つ下の階層のみです。

```
POST /v1/nodes/{parentReferenceId}/folders/bulk
Content-Type: application/json
```

パスパラメータ

パラメータ	型	必須	説明
`parentReferenceId`	string (UUID)	✓	作成先の親フォルダの `reference_id`

リクエストボディ

フィールド	型	必須	説明
`names`	string[]	✓	作成するフォルダ名の配列。1 件以上を指定します。各フォルダ名は最大191文字

- 各フォルダ名には、フォルダ名として使用できない文字（`\ / : * ? " < > |` など）を含められません。`/` も使用できないため、このエンドポイントで階層を指定することはできません（422 エラー）。
- `names` の中に同じ名前を複数指定することはできません（422 エラー）。大文字・小文字は区別されるため、`Alpha` と `ALPHA` は別のフォルダ名として扱われます。
- 既存のフォルダは再利用されず、`names` に指定した名前はすべて新規に作成されます。そのため、指定した親フォルダの直下に同名のファイルまたはフォルダが既に存在する場合は、作成全体が失敗します（409 エラー）。
- **1 リクエストで作成できる件数に専用の上限はありません。** ただし 1 つのフォルダの直下に置けるファイル・フォルダは合計 3,000 件までのため、「既存の子の件数 ＋ `names` の件数」が 3,000 件を超えるリクエストは失敗します（400 エラー）。
- フォルダの作成は**すべて成功するか、1 つも作成されないか**のどちらかです。1 件でもエラーになった場合、それまでに作成されたフォルダも含めてすべて作成されません（ロールバックされます）。
- ただし、この保証はフォルダの作成そのものに限られます。エラーで作成が取り消された場合でも、取り消される前に処理されたフォルダについては、フォルダ内更新通知メールの送信や操作ログの記録が行われていることがあります。

リクエスト例

```
{
  "names": ["契約書", "見積書", "請求書"]
}
```

レスポンス（201 Created）

作成されたフォルダの NodeResource を**配列**で返します。配列は `names` に指定した順に並びます。

```
{
  "data": [
```

```
    {
      "reference_id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
      "name": "契約書",
      "full_path": "/共有フォルダ/営業部/契約書",
      "node_type": "folder",
      ...
    },
    {
      "reference_id": "yyyyyyyy-yyyy-yyyy-yyyy-yyyyyyyyyyyy",
      "name": "見積書",
      "full_path": "/共有フォルダ/営業部/見積書",
      "node_type": "folder",
      ...
    },
    {
      "reference_id": "zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzzzzzz",
      "name": "請求書",
      "full_path": "/共有フォルダ/営業部/請求書",
      "node_type": "folder",
      ...
    }
  ]
}
```

エラーケース

ステータス	コード	説明
422	（バリデーション）	`names` が未指定・配列でない・空の配列、フォルダ名が不正（禁止文字・文字数超過・絵文字・末尾が `.` など）、`names` の中に同じ名前が含まれる
409	`CONFLICT`	同名のファイルまたはフォルダが、指定した親フォルダの直下に既に存在する
400	`ERROR_CODE_NODE_MAX_DEPTH_EXCEEDED`	作成によりフォルダ階層が上限を超える
400	`ERROR_CODE_LIMIT_NODE_CHILDREN`	「既存の子の件数 ＋ `names` の件数」が親フォルダ直下の上限（3,000件）を超える
400	`ERROR_CODE_FULL_PATH`	作成されるフォルダのフルパスが上限（259文字）を超える（フルパスの長さの上限）
403	`ERROR_CODE_PERMISSION_DENIED`	指定した親フォルダへの作成権限がない
423	`ERROR_CODE_COMPANY_NODE_COUNT_LIMIT_REACHED`	「現在のファイル・フォルダ数 ＋ `names` の件数」が契約上の上限を超える
423	`ERROR_CODE_FOLDER_CAPACITY_LIMIT_REACHED`	対象が容量上限に達したフォルダの配下にある（フォルダの容量上限）

```
{
  "error": {
    "code": "CONFLICT",
    "message": "ファイル名またはフォルダ名は既に存在しています。"
  }
}
```

ファイルアップロード（シンプル・15MBまで）

指定したフォルダにファイルをアップロードします。**このエンドポイントは15MB以下のファイル専用**です。

> **【ご注意：サイズ制限】** サーバーの制限により、1リクエストで送信できるのは **15MB まで**です。15MBを超えるファイルは `422` となります。15MBを超えるファイルは、後述のチャンクアップロード（init → chunk → finalize）を使用してください。単体ファイルの最大サイズはチャンクアップロードを利用して **100GB** までです。

```
POST /v1/nodes/{parentReferenceId}/files
Content-Type: multipart/form-data
```

パスパラメータ

パラメータ	型	必須	説明
`parentReferenceId`	string (UUID)	✓	アップロード先のフォルダの `reference_id`

リクエストボディ（multipart/form-data）

フィールド	型	必須	説明
`file`	binary	✓	アップロードするファイル（15MB以下）

レスポンス（201 Created）

アップロードされたファイルの NodeResource を返します。

エラーケース

ステータス	コード	説明
422	（バリデーション）	ファイルサイズが15MBを超えている（チャンクアップロードを使用）
400	`ERROR_CODE_FULL_PATH`	アップロードするファイルのフルパスが上限（259文字）を超える（フルパスの長さの上限）
403	`ERROR_CODE_BULK_COPY_LOCKED`	アップロード先が「まとめてアップロード」の実行中のフォルダ、またはその配下のため操作できない
423	`ERROR_CODE_FOLDER_CAPACITY_LIMIT_REACHED`	対象が容量上限に達したフォルダの配下にある（フォルダの容量上限）
500	`UPLOAD_FAILED`	アップロード処理中にエラーが発生した

チャンクアップロード（大容量ファイル用）

15MBを超えるファイル（最大 **100GB**）は、ファイルを分割して送信するチャンクアップロードを使用します。次の3ステップで行います。

1. **init**: アップロードを開始し、ノードと `upload_id` を発行
2. **chunk**: ファイルを分割した各チャンクを順番に送信（1チャンクは15MB以下）
3. **finalize**: 全チャンク送信後に呼び出してアップロードを確定

> 15MB以下の小さいファイルをチャンクアップロードで送っても構いません（finalize まで正しく実行すれば正常に確定されます）。
> チャンクの分割数は最大 **10000**、S3の制約により末尾以外の各チャンクは **5MB以上** にしてください。

> **【ご注意：レート制限】** チャンク送信（chunk）を含むすべてのリクエストはレート制限の対象です。多数の分割チャンクを並列で一気に送信すると `429 Too Many Requests` となり失敗します。チャンクは順番に（必要に応じて送信間隔を空けて）送信し、レート制限の上限を超えないように制御してください。

① init（アップロード開始）

```
POST /v1/nodes/{parentReferenceId}/files/init
Content-Type: application/json
```

パスパラメータ

パラメータ	型	必須	説明
`parentReferenceId`	string (UUID)	✓	アップロード先のフォルダの `reference_id`

リクエストボディ

フィールド	型	必須	説明
`file_name`	string	✓	ファイル名（最大191文字）

フィールド	型	必須	説明
`size`	integer	✓	ファイルの総バイト数（最大100GB）

レスポンス（201 Created）

```
{
  "data": {
    "reference_id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "upload_id": "（チャンク送信・完了時に使用するアップロードID）"
  }
}
```

エラーケース

ステータス	コード	説明
422	（バリデーション）	`file_name` ／ `size` が不正、またはサイズが100GB超
409	`CONFLICT`	同名のファイルが同じ場所に存在する
400	`ERROR_CODE_FULL_PATH`	アップロードするファイルのフルパスが上限（259文字）を超える（フルパスの長さの上限）。init の時点でエラーになり、アップロードは開始されません
403	`ERROR_CODE_PERMISSION_DENIED`	アップロード先への作成権限がない
403	`ERROR_CODE_BULK_COPY_LOCKED`	アップロード先が「まとめてアップロード」の実行中のフォルダ、またはその配下のため操作できない
423	`ERROR_CODE_*`	ノード数・容量の上限に達している
423	`ERROR_CODE_FOLDER_CAPACITY_LIMIT_REACHED`	対象が容量上限に達したフォルダの配下にある（フォルダの容量上限）

② chunk（チャンク送信）

```
POST /v1/nodes/files/{referenceId}/chunk
Content-Type: multipart/form-data
```

`referenceId` は init で返却された `reference_id` です。チャンクは `chunk_index` の昇順（1から）に送信してください。

リクエストボディ（multipart/form-data）

フィールド	型	必須	説明
`upload_id`	string	✓	init で発行された `upload_id`
`chunk_index`	integer	✓	チャンク番号（1始まり、`total_chunks` 以下）
`total_chunks`	integer	✓	総チャンク数（最大10000）
`chunk`	binary	✓	チャンクデータ（15MB以下。末尾以外は5MB以上）

レスポンス（204 No Content）

エラーケース

ステータス	コード	説明
404	`ERROR_CODE_RESOURCE_NOT_FOUND`	アップロードセッション（`upload_id`）が無効・期限切れ
422	（バリデーション）	パラメータ不正／チャンクが15MB超
403	`ERROR_CODE_BULK_COPY_LOCKED`	アップロード先が「まとめてアップロード」の実行中のフォルダ、またはその配下のため操作できない

> アップロードセッションには有効期限（一定時間）があります。チャンクを送信するたびに期限は延長されますが、長時間放置するとセッションが失効し `404` となります。その場合は init からやり直してください。

③ finalize（アップロード完了）

```
POST /v1/nodes/files/{referenceId}/finalize
```

```
Content-Type: application/json
```

リクエストボディ

フィールド	型	必須	説明
`upload_id`	string	✓	init で発行された `upload_id`
`total_chunks`	integer	✓	送信したチャンクの総数

レスポンス（201 Created）

確定したファイルの NodeResource を返します。

エラーケース

ステータス	コード	説明
404	`ERROR_CODE_RESOURCE_NOT_FOUND`	アップロードセッション（`upload_id`）が無効・期限切れ
400	`UPLOAD_INCOMPLETE`	チャンクの欠落などにより完了できない
403	`ERROR_CODE_BULK_COPY_LOCKED`	アップロード先が「まとめてアップロード」の実行中のフォルダ、またはその配下のため操作できない

ファイルダウンロード（署名付きURL取得）

指定したファイルのダウンロード用URLを取得します。URLには有効期限があります（**省略時は 60 分**）。

> **【ご注意：制限事項】** 外部公開APIのダウンロードは**単一ファイルの同期ダウンロード専用**です。以下の制限があります。
>
> - **ファイルのみ**ダウンロードできます。フォルダを指定した一括ダウンロードはできません（フォルダ指定時は `400` エラー）。
> - ダウンロードできる1ファイルのサイズは **100MB未満** です（100MB以上は `400` エラー）。100MB以上のファイルは、APIではなく **Web または Drive** からダウンロードしてください。

```
POST /v1/nodes/{referenceId}/download
```

パスパラメータ

パラメータ	型	必須	説明
`referenceId`	string (UUID)	✓	ダウンロードするファイルの `reference_id`（ファイルのみ）

リクエストボディ（任意）

パラメータ	型	必須	説明
`expires_minutes`	integer	-	ダウンロードURLの有効期限（分）。`1` 〜 `60` の範囲で指定できます。**省略時は 60 分**。より短い有効期限にしたい場合に指定してください。

レスポンス（200 OK）

```
{
  "data": {
    "download_url": "https://..."
  }
}
```

取得した `download_url` に対してGETリクエストを送ることでファイルをダウンロードできます。URLには有効期限があるため、取得後は速やかに使用してください。

ダウンロードURLの取得は、Web画面からのダウンロードと同様に **操作ログに記録** されます。

エラーケース

ステータス	コード	説明
400	`ERROR_CODE_NOT_AVAILABLE_FOR_DOWNLOAD`	ファイル以外（フォルダ）を指定した
400	`ERROR_CODE_DOWNLOAD_FILE_MAX_SIZE`	ファイルサイズが上限（100MB）以上のため、APIでダウンロードできない
400	`ERROR_CODE_UPLOADING`	指定したファイルがアップロード中
403	`ERROR_CODE_PERMISSION_DENIED`	ダウンロード権限がない、または「マイフォルダを非表示にする」制限が設定されているユーザーがマイフォルダ配下のファイルを指定した
404	`ERROR_CODE_NODE_NOT_FOUND`	指定したノードが存在しない（他社のノードを含む）
404	`ERROR_CODE_NODE_IN_TRASH`	指定したファイルがゴミ箱内にある
423	`ERROR_CODE_PROCESSING_VIRUS_SCAN`	指定したファイルがウイルスチェック中
422	（バリデーション）	`expires_minutes` が整数でない、または `1` 〜 `60` の範囲外

プレビューURL取得

指定したファイルのプレビュー表示用URLを取得します。URLには有効期限があります（**省略時は 5 分**）。

```
GET /v1/nodes/{referenceId}/preview
```

対応ファイル形式

プレビューに対応しているファイル形式と、ファイルサイズの上限は以下のとおりです。上限を超えるファイルはプレビューできません。対応していない拡張子・サイズ超過の場合はエラーになります（「エラーケース」参照）。

種別	対応拡張子	上限サイズ
文書	`doc` `docx` `docm` `xls` `xlsx` `xlsm` `ppt` `pptx` `pptm` `pdf`	10 MB
画像	`png` `jpg` `jpeg` `bmp` `gif` `webp` `svg`	10 MB
テキスト	`txt` `log` `ini` `csv` `tsv` `md`	10 MB
動画	`mp4` `mov` `avi` `wmv`	512 MB
圧縮	`zip`	10 MB

> **ZIPファイルについての注意**
> ZIPファイルの場合、`preview_url` には該当ZIPファイル自体のURLがそのまま返されます（ZIP内のファイル一覧表示などのプレビュー機能は提供されません）。プレビュー表示が必要な場合は、受信者側での実装が必要です。

パスパラメータ

パラメータ	型	必須	説明
`referenceId`	string (UUID)	✓	プレビューするファイルの `reference_id`

クエリパラメータ（任意）

パラメータ	型	必須	説明
`expires_minutes`	integer	-	プレビューURLの有効期限（分）。`1` 〜 `60` の範囲で指定できます。**省略時は 5 分**。

レスポンス（200 OK）

```
{
  "data": {
    "preview_url": "https://..."
  }
}
```

エラーケース

ステータス	コード	説明
422	`PREVIEW_UNAVAILABLE`	このファイルはプレビューに対応していない

ステータス	コード	説明
422	（バリデーション）	`expires_minutes` が整数でない、または `1` 〜 `60` の範囲外
423	`ERROR_CODE_FOLDER_CAPACITY_LIMIT_REACHED`	対象が容量上限に達したフォルダの配下にある（フォルダの容量上限）

リネーム

指定したノード（ファイルまたはフォルダ）の名前を変更します。

```
PUT /v1/nodes/{referenceId}/rename
Content-Type: application/json
```

パスパラメータ

パラメータ	型	必須	説明
`referenceId`	string (UUID)	✓	リネームするノードの `reference_id`

リクエストボディ

フィールド	型	必須	説明
`name`	string	✓	新しい名前（最大191文字）

- 名前には、ファイル名・フォルダ名として使用できない文字（`\ / : * ? " < > |`）と絵文字を含められません。フォルダ名には制御文字も含められません。
- 末尾が `.`（ドット）・タブ・改行の名前、空白だけの名前、Windows の予約名（`CON` `PRN` `AUX` `NUL` `COM1` 〜 `COM9` `LPT1` 〜 `LPT9`。拡張子付きを含む）は指定できません。
- これらはWeb画面でのリネーム、およびフォルダ作成・ファイルアップロードの名前と同じルールです。

リクエスト例

```
{
  "name": "2025年度_改訂版"
}
```

レスポンス（200 OK）

リネーム後のノードの NodeResource を返します。

エラーケース

ステータス	コード	説明
422	（バリデーション）	`name` が未指定、または名前が不正（191文字超・使用できない文字・絵文字・末尾が `.`・予約名など）
404	`ERROR_CODE_NODE_NOT_FOUND`	指定したノードが存在しない（他社のノードを含む）
409	`ERROR_CODE_NODE_DUPLICATE_NAME`	同じフォルダ内に同名のノードが存在する
403	`ERROR_CODE_PERMISSION_DENIED`	対象ノードの編集権限がない、または「マイフォルダを非表示にする」制限が設定されているユーザーがマイフォルダ配下のノードを指定した
403	`ERROR_CODE_NODE_PROTECTED`	対象ノードまたはその配下に、他のユーザーが保護したノードがある
400	`ERROR_CODE_UPLOADING`	対象のファイルがアップロード中
400	`ERROR_CODE_NODE_MAX_DEPTH_EXCEEDED`	対象ノードが階層数の上限を超えた位置にある
423	`ERROR_CODE_COMPANY_STORAGE_LIMIT_REACHED`	契約容量の上限に達している
423	`ERROR_CODE_COMPANY_NODE_COUNT_LIMIT_REACHED`	ファイル・フォルダ数の上限に達している
400	`ERROR_CODE_FULL_PATH`	名前を変更すると、対象のフルパスが上限（259文字）を超える。フォルダの場合は、配下のファイル・フォルダのフルパスも判定する（フルパスの長さの上限）
403	`ERROR_CODE_BULK_COPY_LOCKED`	対象が「まとめてアップロード」の実行中のフォルダ、その配下、またはその上位のフォルダのため操作できない

ノード移動

指定したノード（ファイルまたはフォルダ）を別のフォルダに移動します。

```
PUT /v1/nodes/{referenceId}/move
Content-Type: application/json
```

パスパラメータ

パラメータ	型	必須	説明
`referenceId`	string (UUID)	✓	移動するノードの `reference_id`

リクエストボディ

フィールド	型	必須	説明
`destination_id`	string (UUID)	✓	移動先フォルダの `reference_id`

リクエスト例

```
{
  "destination_id": "zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzzzzzz"
}
```

レスポンス（204 No Content）

エラーケース

ステータス	コード	説明
409	`CONFLICT`	移動先に同名のノードがすでに存在する
400	`ERROR_CODE_FULL_PATH`	移動後のフルパスが上限（259文字）を超える。フォルダの場合は、配下のファイル・フォルダのフルパスも判定する（フルパスの長さの上限）
403	`ERROR_CODE_BULK_COPY_LOCKED`	移動する対象が「まとめてアップロード」の実行中のフォルダ・その配下・その上位のフォルダのいずれか、または移動先が実行中のフォルダ・その配下のため操作できない
423	`ERROR_CODE_FOLDER_CAPACITY_LIMIT_REACHED`	移動先が容量上限に達したフォルダの配下にある（フォルダの容量上限）

```
{
  "error": {
    "code": "CONFLICT",
    "message": "同名のアイテムがすでに存在します。"
  }
}
```

ノードコピー

指定したノード（ファイルまたはフォルダ）を、同じ親フォルダ内に複製します。コピー先に同名のノードがすでに存在する場合は、名前の末尾に `(1)`、`(2)` … のナンバリングが自動で付与されます（例: `報告書.pdf` → `報告書(1).pdf`）。フォルダを指定した場合は、配下のファイル・フォルダもまとめて複製されます。

```
POST /v1/nodes/{referenceId}/copies
```

パスパラメータ

パラメータ	型	必須	説明
`referenceId`	string (UUID)	✓	コピーするノードの `reference_id`

レスポンス（201 Created）

作成されたコピー後のノードの NodeResource を返します。`reference_id` はコピー元とは異なる新しい値になります。

エラーレスポンス

ステータス	エラーコード	条件
400	`ERROR_CODE_FULL_PATH`	コピー後のフルパスが上限（259文字）を超える。名前の末尾に付く `(1)` などを含めて判定し、フォルダの場合は配下のファイル・フォルダのフルパスも判定する（フルパスの長さの上限）
403	`ERROR_CODE_BULK_COPY_LOCKED`	コピー元が「まとめてアップロード」の実行中のフォルダ、またはその配下のため操作できない
422	`ERROR_CODE_COPY_NODE_COUNT_EXCEEDED`	1回のコピーで複製できるファイル・フォルダ数の上限を超えている
422	`ERROR_CODE_COPY_TOTAL_SIZE_EXCEEDED`	1回のコピーで複製できるファイルの合計サイズの上限を超えている
423	`ERROR_CODE_COMPANY_STORAGE_LIMIT_REACHED`	契約容量の上限に達している
423	`ERROR_CODE_COMPANY_NODE_COUNT_LIMIT_REACHED`	ファイル・フォルダ数の上限に達している
423	`ERROR_CODE_FOLDER_CAPACITY_LIMIT_REACHED`	対象が容量上限に達したフォルダの配下にある（フォルダの容量上限）

> **注意:** 1回のコピーで複製できるファイル・フォルダ数には上限があります。上限は、指定したノードとその配下にあるファイル・フォルダの合計件数で判定します（5GB を超えるファイルは、1GB ごとに 1 件として数えます）。上限を超える場合は `422` を返し、コピーは行われません。対象のフォルダを分けて複製してください。
> また、指定したノードとその配下にあるファイルの合計サイズにも上限があります。合計サイズが上限を超える場合も `422` を返し、コピーは行われません。

ノード削除（ゴミ箱へ移動）

指定したノード（ファイルまたはフォルダ）をゴミ箱に移動します。

```
DELETE /v1/nodes/{referenceId}
```

> **注意:** ゴミ箱内のデータは一定期間後に完全削除されます。

パスパラメータ

パラメータ	型	必須	説明
`referenceId`	string (UUID)	✓	削除するノードの `reference_id`

レスポンス（204 No Content）

エラーケース

ステータス	コード	説明
403	`ERROR_CODE_BULK_COPY_LOCKED`	対象が「まとめてアップロード」の実行中のフォルダ、その配下、またはその上位のフォルダのため操作できない

タグ

TagResource（タグオブジェクト）

フィールド	型	説明
`id`	integer	タグのID。タグ操作APIで使用します
`name`	string	タグ名
`is_attached`	boolean	対象ノードにこのタグが付与されているか（ノードのタグ取得時のみ意味を持つ）

タグ作成

新しいタグを作成します。

> **【ご注意：権限制限】** このエンドポイントは、Web の管理画面と同様に**アカウントオーナー権限を持つユーザーのAPIキーのみ**が利用できます。一般ユーザーのAPIキーで実行すると `403` が返ります。

```
POST /v1/tags
Content-Type: application/json
```

リクエストボディ

フィールド	型	必須	説明
`name`	string	✓	タグ名。最大20文字（法人により異なる場合があります）。使用できる文字は英数字・日本語・記号（`-` `_` `#` `@`）・スペース。自社内で同名のタグは作成できません

- 1つの法人で作成できるタグは最大 **100個**（法人により異なる場合があります）までです。上限に達している場合は `400` が返ります。

リクエスト例

```
{
  "name": "重要"
}
```

レスポンス（201 Created）

作成されたタグの TagResource を返します。`is_attached` は作成直後はどのノードにも付与されていないため常に `false` です。

```
{
  "data": {
    "id": 10,
    "name": "重要",
    "is_attached": false
  }
}
```

エラーケース

ステータス	コード	説明
403	`ERROR_CODE_PERMISSION_DENIED`	アカウントオーナー以外のAPIキーで実行した
422	（バリデーション）	`name` が未指定／20文字超（法人により異なる場合があります）／使用不可文字を含む／自社内に同名タグが既存
400	`TAG_LIMIT_EXCEEDED`	タグの作成数が上限（100個。法人により異なる場合があります）に達している

タグ名変更

既存のタグの名前を変更します。

> **【ご注意：権限制限】** このエンドポイントは、**アカウントオーナー権限を持つユーザーのAPIキーのみ**が利用できます。サブオーナー・一般ユーザーのAPIキーで実行すると `403` が返ります。

```
PUT /v1/tags/{tagId}
Content-Type: application/json
```

パスパラメータ

パラメータ	型	必須	説明
`tagId`	integer	✓	名前を変更するタグのID。タグ一覧取得で確認したIDを指定

リクエストボディ

フィールド	型	必須	説明
`name`	string	✓	新しいタグ名。最大20文字（法人により異なる場合があります）。使用できる文字は英数字・日本語・記号（`-` `_` `#` `@`）・スペース。自社内の他のタグと同じ名前には変更できません。現在の名前をそのまま指定しても同名扱いにはなりません

- 名前を変更しても、タグの ID と、タグが付与されているファイル・フォルダは変わりません（付与はそのままで、名前だけが変わります）。
- 変更後は、ノード検索の `tag_name` には新しい名前を指定してください。古い名前では、そのタグが付与されたノードは見つかりません。

リクエスト例

```
{
  "name": "重要案件"
}
```

レスポンス（200 OK）

名前を変更したタグの TagResource を返します。`is_attached` は常に `false` です。

```
{
  "data": {
    "id": 10,
    "name": "重要案件",
    "is_attached": false
  }
}
```

エラーケース

ステータス	コード	説明
422	（バリデーション）	`name` が未指定／文字列以外／空白のみ／20文字超（法人により異なる場合があります）／使用不可文字を含む／自社内の他のタグと同名
403	`ERROR_CODE_PERMISSION_DENIED`	アカウントオーナー以外のAPIキーで実行した
404	`ERROR_CODE_RESOURCE_NOT_FOUND`	指定したタグが存在しない（他社のタグ・削除済みのタグを含む）

タグ削除

タグを削除します。削除したタグは、そのタグが付与されていた全てのファイル・フォルダからも自動的に外れます。

> **【ご注意：権限制限】** このエンドポイントは、Web の管理画面と同様に**アカウントオーナー権限を持つユーザーのAPIキーのみ**が利用できます。一般ユーザーのAPIキーで実行すると `403` が返ります。

```
DELETE /v1/tags/{tagId}
```

パスパラメータ

パラメータ	型	必須	説明
`tagId`	integer	✓	削除するタグのID。タグ一覧取得で確認したIDを指定

レスポンス（204 No Content）

エラーケース

ステータス	コード	説明
403	`ERROR_CODE_PERMISSION_DENIED`	アカウントオーナー以外のAPIキーで実行した
404	`ERROR_CODE_RESOURCE_NOT_FOUND`	指定したタグが存在しない（自社に存在しないIDを含む）

タグ一覧取得

自社で利用可能なタグの一覧を取得します。

```
GET /v1/tags
```

レスポンス（200 OK）

```
{
  "data": [
    {
      "id": 1,
      "name": "重要",
      "is_attached": false
    },
    {
      "id": 2,
      "name": "確認済み",
      "is_attached": false
    }
  ]
}
```

ノードのタグ取得

指定したノードに付与されているタグを、全タグ一覧とともに取得します。

`is_attached: true` のタグが、そのノードに現在付与されています。

```
GET /v1/nodes/{referenceId}/tags
```

パスパラメータ

パラメータ	型	必須	説明
`referenceId`	string (UUID)	✓	タグを取得するノードの `reference_id`

レスポンス（200 OK）

```
{
  "data": [
    {
      "id": 1,
      "name": "重要",
      "is_attached": true
    },
    {
      "id": 2,
      "name": "確認済み",
      "is_attached": false
    }
  ]
}
```

ノードのタグ更新

指定したノードに付与するタグを更新します。

送信した `tag_ids` の配列がそのまま新しいタグとして設定されます（差分更新ではなく**全件上書き**）。タグを全て外す場合は空配列 `[]` を送信してください。

- 1つのノードに設定できるタグは、Web画面と同様に最大 **10個** までです。11個以上を指定した場合は `422` が返ります。
- タグを設定できるのは共有フォルダ配下のノードのみです。マイフォルダ配下のノードには設定できません（`403`）。
- タグの設定は **操作ログに記録** されます。

```
PUT /v1/nodes/{referenceId}/tags
Content-Type: application/json
```

パスパラメータ

パラメータ	型	必須	説明
`referenceId`	string (UUID)	✓	タグを更新するノードの `reference_id`

リクエストボディ

フィールド	型	必須	説明
`tag_ids`	integer[]	✓	付与するタグIDの配列（最大10個）。タグ一覧取得で確認した自社のタグのIDを指定します。同じIDを重複して指定することはできません

リクエスト例（タグを2つ設定）

```
{
  "tag_ids": [1, 2]
}
```

リクエスト例（タグを全て外す）

```
{
  "tag_ids": []
}
```

レスポンス（204 No Content）

エラーケース

ステータス	コード	説明
422	（バリデーション）	`tag_ids` が未指定、11個以上のタグIDを指定した、同じタグIDを重複して指定した、または存在しないタグID（他社のタグ・削除済みのタグを含む）を指定した
404	`ERROR_CODE_NODE_NOT_FOUND`	指定したノードが存在しない（他社のノードを含む）
404	`ERROR_CODE_NODE_IN_TRASH`	指定したノードがゴミ箱内にある
403	`ERROR_CODE_PERMISSION_DENIED`	対象ノードのタグを変更する権限がない
403	`ERROR_CODE_CAN_NOT_UPDATE_TAGS_NODE`	マイフォルダ配下のノードを指定した
403	`ERROR_CODE_NODE_PROTECTED`	対象のファイルが他のユーザーによって保護されている
400	`ERROR_CODE_UPLOADING`	対象のファイルがアップロード中
400	`ERROR_CODE_NODE_MAX_DEPTH_EXCEEDED`	階層数の上限を超えた位置にあるノードに、タグを追加しようとした（タグを外すことはできます）
423	`ERROR_CODE_COMPANY_STORAGE_LIMIT_REACHED`	契約容量の上限に達している
423	`ERROR_CODE_COMPANY_NODE_COUNT_LIMIT_REACHED`	ファイル・フォルダ数の上限に達している
423	`ERROR_CODE_FOLDER_CAPACITY_LIMIT_REACHED`	対象が容量上限に達したフォルダの配下にある（フォルダの容量上限）

お気に入り

お気に入り追加

指定したノードをAPIキーに紐づくユーザーのお気に入りに追加します。

```
PUT /v1/favorites/{referenceId}
```

パスパラメータ

パラメータ	型	必須	説明
`referenceId`	string (UUID)	✓	お気に入りに追加するノードの `reference_id`

- お気に入りに登録できるのは、1ユーザーあたり **300件** までです。
- お気に入りの追加は **操作ログに記録** されます。

レスポンス（204 No Content）

エラーケース

ステータス	コード	説明
404	`ERROR_CODE_NODE_NOT_FOUND`	指定したノードが存在しない、または閲覧権限がない（他社のノードを含む）
404	`ERROR_CODE_NODE_IN_TRASH`	指定したノードがゴミ箱内にある
403	`ERROR_CODE_PERMISSION_DENIED`	「マイフォルダを非表示にする」制限が設定されているユーザーが、マイフォルダまたはその配下のノードを指定した
400	`ERROR_CODE_UPLOADING`	指定したファイルがアップロード中
400	`ERROR_CODE_FAVORITE_EXISTS`	指定したノードはすでにお気に入りに登録されている
400	`ERROR_CODE_MAX_FAVORITE_NODES_REACHED`	お気に入りの登録数が上限（300件）に達している
400	`ERROR_CODE_NODE_MAX_DEPTH_EXCEEDED`	指定したノードが階層数の上限を超えた位置にある

お気に入り削除

指定したノードをお気に入りから削除します。

```
DELETE /v1/favorites/{referenceId}
```

パスパラメータ

パラメータ	型	必須	説明
`referenceId`	string (UUID)	✓	お気に入りから削除するノードの `reference_id`

- お気に入りの削除は **操作ログに記録** されます。

レスポンス（204 No Content）

エラーケース

ステータス	コード	説明
404	`ERROR_CODE_NODE_NOT_FOUND`	指定したノードが存在しない、または閲覧権限がない（他社のノードを含む）
404	`ERROR_CODE_NODE_IN_TRASH`	指定したノードがゴミ箱内にある
404	`ERROR_CODE_RESOURCE_NOT_FOUND`	指定したノードがお気に入りに登録されていない
403	`ERROR_CODE_PERMISSION_DENIED`	「マイフォルダを非表示にする」制限が設定されているユーザーが、マイフォルダまたはその配下のノードを指定した
400	`ERROR_CODE_UPLOADING`	指定したファイルがアップロード中

権限

ファイル・フォルダに対する、ユーザー・グループごとの権限（共有設定）を参照・変更します。

権限レベルは `admin`（管理）・`edit`（編集）・`view`（閲覧）の3種類で、強さは `admin` > `edit` > `view` です。フォルダに設定した権限は、その配下のファイル・フォルダにも継承されます。

- 付与対象に指定する `userId` / `groupId` は、ユーザー・グループのAPIで取得した `id` です。

- **権限の参照・変更ができるのは、対象ノードの権限を変更できる権限（管理権限）を持つ利用者だけです。** 閲覧・編集の権限しか持たない場合は、権限一覧の取得も `403` になります。
- **ベースノード（共有フォルダ・マイフォルダ・ゴミ箱）は権限操作の対象外です。** ベースノード取得で得られる共有フォルダ・マイフォルダの `reference_id` を指定した場合は `403`、ゴミ箱およびゴミ箱内のノードを指定した場合は `404` になります。権限は、共有フォルダ・マイフォルダの配下にあるフォルダ・ファイルに対して設定してください。
- 権限の付与・変更・削除は操作ログに記録されます（操作者はAPIキーに紐づくユーザーになります）。権限一覧の取得は記録されません。
- 契約容量・ファイル数の上限に達している場合、上限のチェックが先に行われるため、存在しないノードや他社のノードを指定しても `404` ではなく `423` が返ります。

NodePermissionResource（権限オブジェクト）

権限系APIのレスポンスに含まれる権限オブジェクトの構造です。ユーザー・グループごとに、継承を含めた**最も強い権限が1件ずつ**返ります。

フィールド	型	説明
`type`	string	対象の種別。`user`（ユーザー）または `group`（グループ）
`id`	integer	対象のユーザーIDまたはグループID
`name`	string	対象のユーザー名またはグループ名
`permission_level_code`	string	権限レベル。`admin`（管理）・`edit`（編集）・`view`（閲覧）
`inherits_permission`	boolean	上位フォルダから継承している権限であれば `true`。対象ノードに直接設定されている権限であれば `false`

- 並び順は**権限レベルの高い順**（`admin` → `edit` → `view`）です。
- **停止中・削除済みのユーザーの権限は含まれません。**
- ユーザーのアバター画像・部署・役職、グループの説明・所属ユーザーは含まれません。

ノードの権限一覧取得

指定したノードに設定されている権限の一覧を取得します。

```
GET /v1/nodes/{referenceId}/permissions
```

パスパラメータ

パラメータ	型	必須	説明
`referenceId`	string (UUID)	✓	権限を取得するノードの `reference_id`

レスポンス（200 OK）

NodePermissionResource の配列を返します。

```
{
  "data": [
    {
      "type": "user",
      "id": 1,
      "name": "山田 太郎",
      "permission_level_code": "admin",
      "inherits_permission": false
    },
    {
      "type": "group",
      "id": 3,
      "name": "営業部",
      "permission_level_code": "view",
```

```
      "inherits_permission": true
    }
  ]
}
```

エラーケース

ステータス	コード	説明
423	`ERROR_CODE_COMPANY_STORAGE_LIMIT_REACHED`	契約容量の上限に達している
423	`ERROR_CODE_COMPANY_NODE_COUNT_LIMIT_REACHED`	ファイル・フォルダ数の上限に達している
423	`ERROR_CODE_FOLDER_CAPACITY_LIMIT_REACHED`	対象が容量上限に達したフォルダの配下にある（フォルダの容量上限）
404	`ERROR_CODE_NODE_NOT_FOUND`	指定したノードが存在しない（他社のノードを含む）
404	`ERROR_CODE_NODE_IN_TRASH`	指定したノードがゴミ箱内にある（ゴミ箱そのものの指定を含む）
403	`ERROR_CODE_PERMISSION_DENIED`	対象ノードの権限を変更できる権限がない（共有フォルダ・マイフォルダのベースノードの指定を含む）
403	`ERROR_CODE_NODE_PROTECTED`	対象ノードまたはその配下に、他のユーザーが保護したノードがある
400	`ERROR_CODE_UPLOADING`	対象のファイルがアップロード中

ユーザーへの権限付与・変更

指定したユーザーに、対象ノードの権限を付与または変更します。すでに権限がある場合は変更、無い場合は新規付与になります。

```
PUT /v1/nodes/{referenceId}/permissions/users/{userId}
Content-Type: application/json
```

パスパラメータ

パラメータ	型	必須	説明
`referenceId`	string (UUID)	✓	権限を設定するノードの `reference_id`
`userId`	integer	✓	対象ユーザーのID。ユーザー一覧取得で確認したIDを指定

リクエストボディ

フィールド	型	必須	説明
`permission_level_code`	string	✓	権限レベル。`admin` ・ `edit` ・ `view` のいずれか

リクエスト例

```
{
  "permission_level_code": "edit"
}
```

レスポンス（200 OK）

設定後の権限一覧を、ノードの権限一覧取得と同じ形式で返します。

```
{
  "data": [
    {
      "type": "user",
      "id": 1,
      "name": "山田 太郎",
      "permission_level_code": "admin",
      "inherits_permission": false
    },
    {
```

```
      "type": "user",
      "id": 25,
      "name": "鈴木 花子",
      "permission_level_code": "edit",
      "inherits_permission": false
    }
  ]
}
```

権限の付与・変更ルール

以下の用語で説明します。権限レベルの強さは `admin` > `edit` > `view` です。

- **新**: リクエストで指定した権限レベル
- **祖先**: 上位フォルダから継承している、同じ対象の最も強い権限
- **現在**: 対象ノードに直接設定されている、同じ対象の権限
- **グループ権限**: 対象ユーザーが所属するグループ経由で適用されている最も強い権限（対象がグループの場合は該当しません）

条件が重なる場合があるため、**A〜D のうち該当する表を選び、上から順に評価して最初に一致した行が結果になります。**

A. 対象ノードに直接権限があり、上位フォルダからの継承もある場合

順	条件	結果
1	新 == 現在	変更せず `200`（何も変わりません）
2	グループ権限があり、新 <= グループ権限	400 `ERROR_CODE_PERMISSION_CAN_NOT_CHANGE`
3	新 == 祖先	400 `ERROR_CODE_PERMISSION_CHANGE_SAME`
4	新 < 祖先	400 `ERROR_CODE_PERMISSION_CHANGE_ANCESTORS`
5	上記以外	変更する（`200`）

B. 対象ノードに直接権限があり、上位フォルダからの継承が無い場合

順	条件	結果
1	新 == 現在	変更せず `200`（何も変わりません）
2	グループ権限があり、新 <= グループ権限	400 `ERROR_CODE_PERMISSION_CAN_NOT_CHANGE`
3	上記以外	変更する（`200`）

C. 対象ノードに直接権限が無く、上位フォルダからの継承がある場合

順	条件	結果
1	新 <= 祖先	400 `ERROR_CODE_PERMISSION_CHANGE_ANCESTORS`
2	グループ権限があり、新 <= グループ権限	400 `ERROR_CODE_PERMISSION_CAN_NOT_GRANT`
3	上記以外	新規付与する（`200`）

D. 対象ノードに直接権限が無く、上位フォルダからの継承も無い場合

順	条件	結果
1	グループ権限があり、新 <= グループ権限	400 `ERROR_CODE_PERMISSION_CAN_NOT_GRANT`
2	上記以外	新規付与する（`200`）

> A-1・B-1 に該当する場合、リクエストは成功（`200`）しますが権限は変更されません。レスポンスの権限一覧で結果を確認してください。指定した権限レベルが現在の権限と同じであれば、上位フォルダから継承している権限より低くても A-1 が先に評価され、`200` になります。
> A-3 のとおり、上位フォルダから継承している権限と同じ権限レベルへは変更できません（変更しても適用される権限が変わらないため）。継承している権限をそのまま適用したい場合は、対象ノードに直接設定された権限を削除してください（ユーザーの権限削除・グループの権限削除）。

そのほかの動作は以下のとおりです。

- 上位の権限を付与すると、その配下にある同じ対象の**下位権限は自動的に削除**されます。
- 操作の結果、対象ノードの管理権限（`admin`）を持つユーザーが1人もいなくなる場合は `400` `ERROR_CODE_PERMISSION_ADMIN_MUST_REMAIN` になります。
- 停止中のユーザーには権限を付与・変更できません（`400` `ERROR_CODE_USER_SUSPENDED`）。

エラーケース

ステータス	コード	説明
423	`ERROR_CODE_COMPANY_STORAGE_LIMIT_REACHED`	契約容量の上限に達している
423	`ERROR_CODE_COMPANY_NODE_COUNT_LIMIT_REACHED`	ファイル・フォルダ数の上限に達している
423	`ERROR_CODE_FOLDER_CAPACITY_LIMIT_REACHED`	対象が容量上限に達したフォルダの配下にある（フォルダの容量上限）
404	`ERROR_CODE_NODE_NOT_FOUND`	指定したノードが存在しない（他社のノードを含む）
404	`ERROR_CODE_NODE_IN_TRASH`	指定したノードがゴミ箱内にある
404	`ERROR_CODE_RESOURCE_NOT_FOUND`	指定したユーザーが存在しない（他社・削除済みを含む）
403	`ERROR_CODE_PERMISSION_DENIED`	対象ノードの権限を変更できる権限がない、またはファイル・フォルダの共有が制限されている
403	`ERROR_CODE_NODE_PROTECTED`	対象ノードまたはその配下に、他のユーザーが保護したノードがある
400	`ERROR_CODE_UPLOADING`	対象のファイルがアップロード中
400	`ERROR_CODE_NODE_MAX_DEPTH_EXCEEDED`	対象ノードが階層数の上限を超えた位置にある
400	`ERROR_CODE_USER_SUSPENDED`	対象ユーザーが停止中
400	`ERROR_CODE_PERMISSION_CHANGE_SAME`	上位フォルダから継承している権限と同じ権限へ変更しようとした（ルール A-3）
400	`ERROR_CODE_PERMISSION_CHANGE_ANCESTORS`	上位フォルダから継承している権限より低い権限へ変更しようとした（ルール A-4）、または継承している権限と同じか、それより低い権限を新規付与しようとした（ルール C-1）
400	`ERROR_CODE_PERMISSION_CAN_NOT_CHANGE`	所属グループ経由の権限と同じか、それより低い権限へ変更しようとした（ルール A-2 / B-2）
400	`ERROR_CODE_PERMISSION_CAN_NOT_GRANT`	所属グループ経由の権限と同じか、それより低い権限を新規付与しようとした（ルール C-2 / D-1）
400	`ERROR_CODE_PERMISSION_ADMIN_MUST_REMAIN`	変更すると対象ノードの管理権限を持つユーザーがいなくなる
422	（バリデーション）	`permission_level_code` が未指定、または `admin` ・ `edit` ・ `view` 以外

グループへの権限付与・変更

指定したグループに、対象ノードの権限を付与または変更します。

```
PUT /v1/nodes/{referenceId}/permissions/groups/{groupId}
Content-Type: application/json
```

パスパラメータ

パラメータ	型	必須	説明
`referenceId`	string (UUID)	✓	権限を設定するノードの `reference_id`
`groupId`	integer	✓	対象グループのID。グループ一覧取得で確認したIDを指定

リクエストボディ・レスポンス・付与ルール・エラーケースはユーザーへの権限付与・変更と同じです。対象がグループになる点だけが異なります。

- 対象がグループのため、付与ルールの「グループ権限」の条件には該当しません（`ERROR_CODE_PERMISSION_CAN_NOT_CHANGE` / `ERROR_CODE_PERMISSION_CAN_NOT_GRANT` は発生しません）。同じ理由で `ERROR_CODE_USER_SUSPENDED` も発生しません（グループに停止中のユーザーが含まれていても付与できます）。
- 対象グループが存在しない場合（他社・削除済みを含む）は `404` `ERROR_CODE_RESOURCE_NOT_FOUND` になります。
- 保護グループ（「すべてのユーザー」）にも権限を付与できます。

ユーザーの権限削除

指定したユーザーの、対象ノードに直接設定された権限を削除します。

```
DELETE /v1/nodes/{referenceId}/permissions/users/{userId}
```

> **【ご注意：削除は2段階です】** `accept_delete` を指定しない最初のリクエストは、**削除しても権限が残る可能性などを知らせる確認の警告（`400`）を返すだけで、権限は削除されません。** 内容を確認したうえで、同じリクエストに `accept_delete=true` を付けて再度呼び出すと削除が実行されます。

パスパラメータ

パラメータ	型	必須	説明
`referenceId`	string (UUID)	✓	権限を削除するノードの `reference_id`
`userId`	integer	✓	対象ユーザーのID

クエリパラメータ

パラメータ	型	必須	説明
`accept_delete`	boolean	-	削除を承認する場合に `true`（`1` も可）を指定します。**省略時は `false` となり、確認の警告のみが返ります**

リクエスト例

```
# 1回目：確認（削除されません。400 と警告内容が返ります）
DELETE /v1/nodes/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/permissions/users/25

# 2回目：承認して削除
DELETE /v1/nodes/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/permissions/users/25?accept_delete=true
```

レスポンス（200 OK）

削除後の権限一覧を、ノードの権限一覧取得と同じ形式で返します。

削除ルール

対象ノードへの権限の付き方によって、返る警告のコードと `accept_delete=true` を付けたときの結果が変わります。

対象ノードの直接権限	上位フォルダからの継承	所属グループ経由	`accept_delete` なし	`accept_delete=true`
あり	あり	–	400 `ERROR_CODE_PERMISSION_STILL_REMAIN`	**削除される（200）**
あり	なし	あり	400 `ERROR_CODE_PERMISSION_GROUP_USER_DELETE_SAME`	**削除される（200）**
あり	なし	なし	400 `ERROR_CODE_PERMISSION_DELETE`	**削除される（200）**
なし	あり	–	400 `ERROR_CODE_PERMISSION_CHANGE_ANCESTORS`	400 `ERROR_CODE_PERMISSION_CAN_NOT_DELETE`
なし	なし	–	400 `ERROR_CODE_PERMISSION_CAN_NOT_DELETE`	400 `ERROR_CODE_PERMISSION_CAN_NOT_DELETE`

> **継承している権限は、この階層からは削除できません。** 対象ノードに直接設定された権限が無く、上位フォルダから継承しているだけの場合は、`accept_delete=true` を指定しても削除できません。権限一覧で `inherits_permission` が `true` の権限がこれに該当します。削除するには、その権限が直接設定されている**上位フォルダ側**で削除してください。
>
> `ERROR_CODE_PERMISSION_STILL_REMAIN` / `ERROR_CODE_PERMISSION_GROUP_USER_DELETE_SAME` は、直接設定された権限を削除しても、上位フォルダからの継承や所属グループ経由の権限が残ることを知らせる警告です。`accept_delete=true` で削除を実行しても、これらの経路による権限は残ります。

エラーケース

ステータス	コード	説明
423	`ERROR_CODE_COMPANY_STORAGE_LIMIT_REACHED`	契約容量の上限に達している
423	`ERROR_CODE_COMPANY_NODE_COUNT_LIMIT_REACHED`	ファイル・フォルダ数の上限に達している

ステータス	コード	説明
423	`ERROR_CODE_FOLDER_CAPACITY_LIMIT_REACHED`	対象が容量上限に達したフォルダの配下にある（フォルダの容量上限）
404	`ERROR_CODE_NODE_NOT_FOUND`	指定したノードが存在しない（他社のノードを含む）
404	`ERROR_CODE_NODE_IN_TRASH`	指定したノードがゴミ箱内にある
404	`ERROR_CODE_RESOURCE_NOT_FOUND`	指定したユーザーが存在しない（他社・削除済みを含む）
403	`ERROR_CODE_PERMISSION_DENIED`	対象ノードの権限を変更できる権限がない、またはファイル・フォルダの共有が制限されている
403	`ERROR_CODE_NODE_PROTECTED`	対象ノードまたはその配下に、他のユーザーが保護したノードがある
400	`ERROR_CODE_UPLOADING`	対象のファイルがアップロード中
400	`ERROR_CODE_PERMISSION_ADMIN_MUST_REMAIN`	削除すると対象ノードの管理権限を持つユーザーがいなくなる
400	`ERROR_CODE_PERMISSION_STILL_REMAIN`	削除の確認。削除しても上位フォルダから継承している権限が残る
400	`ERROR_CODE_PERMISSION_GROUP_USER_DELETE_SAME`	削除の確認。削除しても所属グループ経由の権限が残る
400	`ERROR_CODE_PERMISSION_DELETE`	削除の確認。対象ノードに直接設定された権限を削除してよいか
400	`ERROR_CODE_PERMISSION_CHANGE_ANCESTORS`	`accept_delete` を指定せずに、継承しているだけの権限を削除しようとした
400	`ERROR_CODE_PERMISSION_CAN_NOT_DELETE`	対象ノードに削除できる権限が直接設定されていない
422	（バリデーション）	`accept_delete` が真偽値として解釈できない

グループの権限削除

指定したグループの、対象ノードに直接設定された権限を削除します。

```
DELETE /v1/nodes/{referenceId}/permissions/groups/{groupId}
```

パスパラメータ

パラメータ	型	必須	説明
`referenceId`	string (UUID)	✓	権限を削除するノードの `reference_id`
`groupId`	integer	✓	対象グループのID

クエリパラメータ・レスポンス・削除ルール・エラーケースはユーザーの権限削除と同じです。対象がグループになる点だけが異なります（削除も2段階で、`accept_delete` の指定が必要です）。

- 対象がグループのため、削除ルールの「所属グループ経由」の条件には該当しません（`ERROR_CODE_PERMISSION_GROUP_USER_DELETE_SAME` は発生しません）。
- 対象グループが存在しない場合（他社・削除済みを含む）は `404` `ERROR_CODE_RESOURCE_NOT_FOUND` になります。

ユーザー・グループ

権限の付与対象を選ぶための、自社のユーザー・グループの参照APIと、グループの管理API（作成・更新・削除・所属ユーザーの追加・削除）です。

参照系の3本（ユーザー一覧取得・グループ一覧取得・グループ所属ユーザー取得）は、一般ユーザーのAPIキーでも利用できます（権限の付与対象を選べないと権限設定が行えないため）。
グループの作成・更新・削除・所属ユーザーの追加・削除は、アカウントオーナーのAPIキーのみが利用できます（各エンドポイントの注記を参照）。

UserSummaryResource（ユーザーオブジェクト）

フィールド	型	説明
`id`	integer	ユーザーID。権限付与APIの `userId` に指定します
`name`	string	ユーザー名
`department_name`	`string | null`	所属名（未設定の場合は `null`）
`position_name`	`string | null`	役職名（未設定の場合は `null`）

> ログインID・メールアドレス・ユーザー種別・ユーザーの状態・アバター画像は返しません。

ユーザー一覧取得

自社のユーザー一覧を取得します。権限の付与対象を選ぶために使用します。

```
GET /v1/users
```

- **有効なユーザーのみ**が対象です。停止中・削除済みのユーザーは含まれません。
- APIキーに紐づくユーザー自身も含まれます。
- 並び順はユーザーIDの昇順です。
- 全件を返します。

レスポンス（200 OK）

UserSummaryResource の配列を返します。

```
{
  "data": [
    {
      "id": 1,
      "name": "山田 太郎",
      "department_name": "営業部",
      "position_name": "部長"
    },
    {
      "id": 25,
      "name": "山田 花子",
      "department_name": null,
      "position_name": null
    }
  ]
}
```

GroupResource（グループオブジェクト）

フィールド	型	説明
`id`	integer	グループID。権限付与APIの `groupId` に指定します
`name`	string	グループ名
`description`	`string | null`	グループの説明（未設定の場合は `null`）
`users_count`	integer	所属ユーザー数。**停止中のユーザーも含みます**（削除済みのユーザー・解除済みの所属は含みません）
`is_protected`	boolean	保護グループ（「すべてのユーザー」）かどうか。`true` の場合は編集・削除ができないグループです
`created_at`	string	作成日時（ISO 8601形式）
`updated_at`	string	更新日時（ISO 8601形式）

- `users_count` は、グループ所属ユーザー取得が返す件数と必ず一致します。
- グループの作成者は返しません。

グループ一覧取得

自社のグループ一覧を取得します。

```
GET /v1/groups
```

- 保護グループ「すべてのユーザー」も含めた**全グループ**を返します（`is_protected` で判別できます）。
- 並び順はグループIDの昇順です。
- 全件を返します。

レスポンス（200 OK）

GroupResource の配列を返します。


```
{
  "data": [
    {
      "id": 1,
      "name": "すべてのユーザー",
      "description": null,
      "users_count": 120,
      "is_protected": true,
      "created_at": "2025-01-01T00:00:00+09:00",
      "updated_at": "2025-01-01T00:00:00+09:00"
    },
    {
      "id": 3,
      "name": "営業部",
      "description": "営業部のメンバー",
      "users_count": 12,
      "is_protected": false,
      "created_at": "2025-02-01T10:00:00+09:00",
      "updated_at": "2025-04-01T12:00:00+09:00"
    }
  ]
}
```

グループ所属ユーザー取得

指定したグループに所属しているユーザーの一覧を取得します。

```
GET /v1/groups/{groupId}/users
```

- **停止中のユーザーも含みます**（削除済みのユーザー・解除済みの所属は含みません）。そのため件数はグループ一覧取得の `users_count` と必ず一致します。
- 保護グループ「すべてのユーザー」も対象です。
- 並び順・返却項目はユーザー一覧取得と同じです（ユーザーIDの昇順）。
- 全件を返します。

> 権限の付与対象を選ぶ用途には、停止中のユーザーを含まないユーザー一覧取得をご利用ください。

パスパラメータ

パラメータ	型	必須	説明
`groupId`	integer	✓	所属ユーザーを取得するグループのID

レスポンス（200 OK）

UserSummaryResource の配列を返します。


```
{
  "data": [
    {
```

```
      "id": 1,
      "name": "山田 太郎",
      "department_name": "営業部",
      "position_name": "部長"
    }
  ]
}
```

エラーケース

ステータス	コード	説明
404	`ERROR_CODE_RESOURCE_NOT_FOUND`	指定したグループが存在しない（他社のグループ・削除済みのグループを含む）

グループ作成

グループを作成します。作成と同時に所属ユーザーも指定できます。

> **【ご注意：権限制限】** このエンドポイントは、Web の管理画面と同様に**アカウントオーナー権限を持つユーザーのAPIキーのみ**が利用できます。サブオーナー・一般ユーザーのAPIキーで実行すると `403` が返ります。

```
POST /v1/groups
Content-Type: application/json
```

リクエストボディ

フィールド	型	必須	説明
`name`	string	✓	グループ名。最大20文字。使用できる文字は英数字・日本語・記号（`-` `_` `#` `@`）・スペース。空白のみは指定できません。自社内で同名のグループは作成できません（下記「グループ名の制約」を参照）
`description`	string	–	グループの説明。最大200文字。絵文字は使用できません。未指定・`null`・空文字はいずれも「説明なし（`null`）」として扱われます
`user_ids`	integer[]	–	作成と同時に所属させるユーザーIDの配列。自社のユーザー（**停止中のユーザーを含む**）を指定できます。重複したIDは1件として扱われます。空配列も指定できます

グループ名の制約

- 自社内でユニークである必要があります。保護グループ「すべてのユーザー」と同じ名前も指定できません。
- 同名かどうかは**完全一致**で判定します（大文字・小文字、全角・半角を区別します）。
- 削除済みのグループと同じ名前は使用できます。
- 同名のグループを同時に（並行して）作成した場合、後から実行された側は**既存グループへの `user_ids` の追加として扱われることがあります**（その場合も `201` で既存グループが返ります）。

その他の仕様

- 1つの法人で作成できるグループは最大 **100件**（保護グループ「すべてのユーザー」を含む）までです。上限に達している場合は `400` が返ります。
- `user_ids` に自社に存在しないユーザー（他社のユーザー・存在しないID・削除済みのユーザー）が含まれている場合、該当するIDは除外して登録します。指定したIDの**すべて**が該当する場合は `404` が返り、グループは作成されません。
- 文字列の前後の空白は除去してから検証・保存します。空文字は未指定として扱われます。

リクエスト例

```
{
  "name": "営業部",
  "description": "営業部のメンバー",
  "user_ids": [1, 25]
}
```

レスポンス（201 Created）

作成されたグループの GroupResource を返します。`users_count` は応答時点の所属ユーザー数です（新規作成の場合は `user_ids` で実際に登録された人数と一致します。上記「グループ名の制約」の並行作成により既存グループが返った場合は、そのグループの所属ユーザー数になります）。

```
{
  "data": {
    "id": 3,
    "name": "営業部",
    "description": "営業部のメンバー",
    "users_count": 2,
    "is_protected": false,
    "created_at": "2025-02-01T10:00:00+09:00",
    "updated_at": "2025-02-01T10:00:00+09:00"
  }
}
```

エラーケース

ステータス	コード	説明
422	（バリデーション）	`name` が未指定／空白のみ／20文字超／使用不可文字を含む／自社内に同名グループが既存（保護グループ「すべてのユーザー」を含む）。`description` が200文字超／絵文字を含む
403	`ERROR_CODE_PERMISSION_DENIED`	アカウントオーナー以外のAPIキーで実行した
400	`ERROR_CODE_GROUP_LIMIT_REACHED`	グループの作成数が上限（100件。保護グループを含む）に達している
404	`ERROR_CODE_RESOURCE_NOT_FOUND`	`user_ids` に指定したすべてのユーザーが自社に存在しない（他社・存在しないID・削除済みを含む）。グループは作成されません

グループ更新

グループの名前・説明を更新します。**全項目の置き換え**であり、部分更新ではありません。

> **【ご注意：権限制限】** このエンドポイントは、Web の管理画面と同様に**アカウントオーナー権限を持つユーザーのAPIキーのみ**が利用できます。サブオーナー・一般ユーザーのAPIキーで実行すると `403` が返ります。

```
PUT /v1/groups/{groupId}
Content-Type: application/json
```

パスパラメータ

パラメータ	型	必須	説明
`groupId`	integer	✓	更新するグループのID。グループ一覧取得で確認したIDを指定

リクエストボディ

フィールド	型	必須	説明
`name`	string	✓	グループ名。制約はグループ作成と同じです。**現在の名前をそのまま指定しても同名扱いにはなりません**
`description`	string	–	グループの説明。制約はグループ作成と同じです。**省略・`null`・空文字はいずれも説明を空（`null`）にします**

- 保護グループ「すべてのユーザー」は更新できません（`400`）。
- 現在と同じ内容を送っても `200` で成功します。
- 文字列の前後の空白は除去してから検証・保存します。

リクエスト例

```
{
  "name": "営業部",
  "description": "営業部のメンバー（東日本）"
}
```

レスポンス（200 OK）

更新後のグループの GroupResource を返します。`users_count` は応答時点の所属ユーザー数です。

```
{
  "data": {
    "id": 3,
    "name": "営業部",
    "description": "営業部のメンバー（東日本）",
    "users_count": 12,
    "is_protected": false,
    "created_at": "2025-02-01T10:00:00+09:00",
    "updated_at": "2025-04-01T12:00:00+09:00"
  }
}
```

エラーケース

ステータス	コード	説明
422	（バリデーション）	`name` が未指定／空白のみ／20文字超／使用不可文字を含む／自社内の他のグループと同名。`description` が200文字超／絵文字を含む
403	`ERROR_CODE_PERMISSION_DENIED`	アカウントオーナー以外のAPIキーで実行した
404	`ERROR_CODE_RESOURCE_NOT_FOUND`	指定したグループが存在しない（他社のグループ・削除済みのグループを含む）
400	`ERROR_CODE_GROUP_PROTECTED`	保護グループ「すべてのユーザー」を指定した

グループ削除

グループを削除します。

> **【ご注意：権限制限】** このエンドポイントは、Web の管理画面と同様に**アカウントオーナー権限を持つユーザーのAPIキーのみ**が利用できます。サブオーナー・一般ユーザーのAPIキーで実行すると `403` が返ります。

```
DELETE /v1/groups/{groupId}
```

パスパラメータ

パラメータ	型	必須	説明
`groupId`	integer	✓	削除するグループのID。グループ一覧取得で確認したIDを指定

削除に伴う挙動（Web の管理画面と同じ）

- 所属ユーザーの紐づけはすべて解除されます。
- そのグループに付与されていた権限は、ノードの権限一覧取得に表示されなくなり、所属していたユーザーはグループ経由で得ていた権限を失います。
- そのグループが外部共有リンクの「リンクの発行申請」の「申請が必須になるグループ・ユーザー」に設定されている場合、申請の有効・無効を問わず対象から自動的に外れます。ただし、**「リンクの発行申請」が有効で、かつそのグループが唯一の対象**に設定されている場合は削除できず `422` が返ります（下記エラーケースを参照）。
- 保護グループ「すべてのユーザー」は削除できません（`400`）。

レスポンス（204 No Content）

エラーケース

ステータス	コード	説明
403	`ERROR_CODE_PERMISSION_DENIED`	アカウントオーナー以外のAPIキーで実行した
404	`ERROR_CODE_RESOURCE_NOT_FOUND`	指定したグループが存在しない（他社のグループ・削除済みのグループを含む）
400	`ERROR_CODE_GROUP_PROTECTED`	保護グループ「すべてのユーザー」を指定した
422	`EXTERNAL_SHARE_LINK_APPROVAL_SETTING_DELETION_DENIED`	「リンクの発行申請」が有効で、指定したグループが唯一の「申請が必須になるグループ・ユーザー」に設定されている。グループは削除されません。**この 422 は `errors` 形式で返ります**

グループへのユーザー追加

グループにユーザーを1人追加します。リクエストボディはありません。

> **【ご注意：権限制限】** このエンドポイントは、Web の管理画面と同様に**アカウントオーナー権限を持つユーザーのAPIキーのみ**が利用できます。サブオーナー・一般ユーザーのAPIキーで実行すると `403` が返ります。

```
PUT /v1/groups/{groupId}/users/{userId}
```

パスパラメータ

パラメータ	型	必須	説明
`groupId`	integer	✓	追加先のグループのID
`userId`	integer	✓	追加するユーザーのID。自社のユーザー（**停止中のユーザーを含む**）を指定できます

- 1リクエストで追加できるのは1人です。複数人をまとめて追加する場合は、グループ作成の `user_ids` をご利用ください。
- **冪等**です。既に所属しているユーザーを指定しても `200` が返り、所属は変わりません（二重登録にはなりません）。
- 過去に所属を解除されたユーザーも再度追加できます。
- 保護グループ「すべてのユーザー」には追加できません（`400`）。

レスポンス（200 OK）

操作後の**所属ユーザー全員**を返します。内容・並び順はグループ所属ユーザー取得と同じです（UserSummaryResource の配列、ユーザーIDの昇順）。


```
{
  "data": [
    {
      "id": 1,
      "name": "山田 太郎",
      "department_name": "営業部",
      "position_name": "部長"
    },
    {
      "id": 25,
      "name": "山田 花子",
      "department_name": null,
      "position_name": null
    }
  ]
}
```


エラーケース

ステータス	コード	説明
403	`ERROR_CODE_PERMISSION_DENIED`	アカウントオーナー以外のAPIキーで実行した

ステータス	コード	説明
404	`ERROR_CODE_RESOURCE_NOT_FOUND`	指定したグループが存在しない（他社のグループ・削除済みのグループを含む）
400	`ERROR_CODE_GROUP_PROTECTED`	保護グループ「すべてのユーザー」を指定した
404	`ERROR_CODE_RESOURCE_NOT_FOUND`	指定したユーザーが存在しない（他社のユーザー・削除済みのユーザーを含む）

グループからのユーザー削除

グループからユーザーを1人外します。リクエストボディはありません。

> **【ご注意：権限制限】** このエンドポイントは、Web の管理画面と同様に**アカウントオーナー権限を持つユーザーのAPIキーのみ**が利用できます。サブオーナー・一般ユーザーのAPIキーで実行すると `403` が返ります。

```
DELETE /v1/groups/{groupId}/users/{userId}
```

パスパラメータ

パラメータ	型	必須	説明
`groupId`	integer	✓	対象のグループのID
`userId`	integer	✓	外すユーザーのID。自社のユーザー（**停止中のユーザーを含む**）を指定できます

- 1リクエストで外せるのは1人です。
- **冪等**です。所属していないユーザーを指定しても `200` が返り、何も変わりません。
- 外したユーザーは、そのグループ経由で得ていた権限を失います（グループ自体の権限は残ります）。
- 保護グループ「すべてのユーザー」からは外せません（`400`）。

レスポンス（200 OK）

操作後の**所属ユーザー全員**を返します。内容・並び順はグループへのユーザー追加と同じです。

```
{
  "data": [
    {
      "id": 1,
      "name": "山田 太郎",
      "department_name": "営業部",
      "position_name": "部長"
    }
  ]
}
```

エラーケース

ステータス	コード	説明
403	`ERROR_CODE_PERMISSION_DENIED`	アカウントオーナー以外のAPIキーで実行した
404	`ERROR_CODE_RESOURCE_NOT_FOUND`	指定したグループが存在しない（他社のグループ・削除済みのグループを含む）
400	`ERROR_CODE_GROUP_PROTECTED`	保護グループ「すべてのユーザー」を指定した
404	`ERROR_CODE_RESOURCE_NOT_FOUND`	指定したユーザーが存在しない（他社のユーザー・削除済みのユーザーを含む）