\Orchestrating a brighter world NEC

モバイルバックエンド基盤
REST API リファレンス

Ver 7.0.2

2017 年 12 月 22 日

日本電気株式会社

改版履歴

版数	日付	改版内容
1.0.0	2015/1/22	● SDE Push 登録レスポンス変更 ● インスタレーションの versionCode の型を整数に変更 ● ログイン時の username と email の条件を明確化 ● Push 送信時の JSON に content-available を追加。 ● 残 TBD 項目を確定 ● サインアップ時に username を指定しない場合はランダム値が設定される旨を記載 ● 5.1 サインアップに、同一ユーザの説明を追記 ● インスタレーション：不要な BadRequest を削除
1.0.1	2015/3/4	● 3.2 ACL 曖昧な部分を修正。owner が存在しない場合はフィールド自体が無い旨を記載。 ● 5.1 サインアップ、5.4 ユーザ情報変更、6.1 グループ作成・変更の注意事項に、username,email,password,グループ名の各ポリシを記載 ● 5.6 パスワードリセット要求 ・ リセット要求のインターバルの値は 10 分であることを記載 ・ リクエストリミットを超えた場合のレスポンスコードを 403 エラーに修正 ・ 404 エラーを追記 ・ 200 OK について、「パスワードリセットメールを正常に送信した」に記載を修正 ● 5.7 ユーザ情報の検索 404 エラーを追記 ● 5.9 自ユーザ情報の取得 403 エラーは返さないので記載を削除 ● 7.1 バケットの作成・更新 バケット名の長さ制限について記載 ● 8.1 オブジェクトの作成 フィールド名制限について修正 ● 8.3 オブジェクトクエリ パラメータの曖昧な記載を修正。count は検索条件に合致する全件数である旨を記載。 ● 8.5 削除マークに関する記述を修正 ● 9.1 ファイルの新規アップロード ファイル名の長さ制限について修正 ● 10.1 SDE4SD Commons Push を使用する場合のレスポンスに uri を追加。登録 API に再登録処理について追記。
1.0.2	2015/3/9	● 8.3 オブジェクトクエリ時に currentTime を返却する機能を追加
1.0.3	2015/5/11	● 5. ユーザ管理(ユーザ操作)、 6. ユーザ管理(グループ管理系) 各 API について、ETag を追記 ● 5.1 サインアップ, 5.4 ユーザ情報変更, 5.7 ユーザ情報の検索, 5.8 ユーザ情報の取得, 5.9 自ユーザ情報の取得 レスポンス内容を記載。また、5.7 ユーザ情報の検索 では、”groups”フィールドを含まない旨を記載。 ● 5.5 ユーザ削除, 6.4 グループ削除 所属するグループからも削除される旨を追記。 ● 5.6 パスワードリセット要求

		レスポンスコードに「401 Unauthorized： 認証エラー」が記載されていないので追記。権限エラーは発生しないので記載を削除。 ● 6.1 グループ作成・変更 登録済みのユーザ、グループを指定する必要がある旨を追記。 users, groups をオプションに変更。また、指定しない場合の挙動を追記。 自分が所属するグループを確認する方法を記載 ● 6.2 グループ一覧の取得, 6.3 グループの取得, 6.4 グループ削除 400 エラーは返さないのでレスポンスコードから削除 ● 9.1 ファイルの新規アップロード reasonCode に対する detail の内容を追記。 レスポンスコードに「ファイル(論理削除データ)がロック中（file_locked）」を追記 ● 9.3 ファイルダウンロード ➢ 発生しないレスポンスコードを削除（400 Bad Request） ● 9.8 ファイルの公開 ➢ 公開中のファイルの再公開に関する曖昧表現の修正 ● 10.3 インスタレーション更新 ➢ インスタレーション不正更新時の注意事項を追記
1.0.4	2015/6/5	● 4.4 エラー通知 ➢ 共通エラー定義を追加 ● 5. ユーザ管理(ユーザ操作) ● 5.1 サインアップ ➢ username, password の文字種制限削除 ● 6. ユーザ管理(グループ管理系) ➢ 各 API のレスポンスに、ETag を追記 ● 5.4 ユーザ情報変更(パスワード変更を含む) ➢ option フィールドに任意のオブジェクトを登録できる旨を追記。それに伴い「5. ユーザ管理」の各 API のレスポンスに option フィールドを追記 ● 6.5 所属ユーザ・グループの追加 ● 6.6 所属ユーザ・グループの削除 ➢ 新規 API を作成 ● 5.4 ユーザ情報変更 ● 5.5 ユーザ削除 ● 5.6 パスワードリセット要求 ● 6.1 グループ作成・変更 ● 6.4 グループ削除 ● 6.5 所属ユーザ・グループの追加 ● 6.6 所属ユーザ・グループの削除 ➢ 楽観ロックによる Conflict 発生時のレスポンスを追記 ● 8.6 オブジェクトの一括削除を追加 ● 10.6 allowedReceivers について追記 ● 7 バケット管理からインデックス・シャードキー 関連の REST API を削除
1.0.5	2015/6/19	● 5.4, 5.5 ユーザ情報変更・削除の際のセッショントークン要件を修正(マスターキー使用時は不要) ● 8.7 バッチリクエストの requestToken を用いたレスポンスキャッシュについて注意事項を追記 ● 5. ユーザ管理(ユーザ操作) ➢ 各 API のレスポンスに最終ログイン日時(lastLoginAt)を追加 ➢ LDAP 認証を使う場合は機能を使えない旨を記載 ● 5.2 ログイン

		➢ LDAP 認証後に MongoDB へ保存するユーザ情報の内容を記載
1.0.6	2015/8/27	● 3 認証の記述を追加 ● 4.2 JSON プロパティ名の予約語を追加 ● 4.3 日付表記について補足追加 ● 4.5 HTTP ヘッダの記述を追加 ● 4.6 クエリパラメータの記述を追加 ● 7 バケット管理 のレスポンスコードの記述を修正 ● 8.3.5 プロジェクションを追加 ● 8.2 削除マークについての記述を修正 ● 8.7 バッチリクエスト etag の記述を修正 ● 8.3.2 ソート順序に注意事項を追記
3.0.0	2015/11/11	● 8.7 バッチリクエストのデータ注記を追記 ● push のグループ指定に関する説明を追加 ● 9.1 と 9.2 にエラーコード追記（507, Insufficient Storage） ● 「仮削除」を「論理削除」に文言修正 ● 5.1 サインアップ、6.1 グループ作成・変更について、ポリシ違反の場合は 400 エラーを返却する旨を追記 ● 5.2 ログインの expire のデフォルトがログインから 24 時間後である旨を追記 ● 5.2 ログインの LDAP 認証で email を指定しても無視される旨を追記 ● 6.1 グループ名のポリシに関する記載が重複していたので、片方を削除 ● 8 オブジェクト操作時の 500InternalServerError について追記 ● 8 部分更新について説明を修正 ● 8.3 オブジェクトソート時の容量制限を追記 ● 8.7 バッチオペレーションの結果に serverError を追加。レスポンスの説明を修正。 ● 9.2 ファイルの更新アップロードには、対象ファイルの update 権限も必要の旨を追記 ● 6.1, 7.1, 8.4, 9.2, 9.6, 9.8, 9.9 グループ、バケット、オブジェクト、ファイルの更新時に、read 権限の有無に関わらず同じレスポンス内容が返却される旨を追記。 ● 9.3 ファイルのダウンロードのレスポンスコードに Conflict の記載を追記。
3.0.1	2015/12/16	● 2.2 ACL の詳細を追記
4.0.0	2016/1/27	● 10.1 SDE Push についての記述を削除 ● 10.1 ・SSE Push についての記述を追加・修正 ・TBD 部分を削除 ● 10.6 SSE Push についての記述を追加 ● 「6.1 グループの作成」を追加 ● 7.4 バケット削除に、バケット強制削除の旨を追記 ● 10.1 インスタレーションの _osType に "java" を追加 ● 8.7 バッチオペレーションの処理順について追記
4.0.1	2016/3/1	● 8.3 オブジェクトクエリの readPreference 指定に対応 ● 8.1 Conflict 発生時の detail フィールドに関する記載ミスを修正
5.0.0	2016/9/13	● 8.4 オブジェクトのロングクエリ API を追加 ● 8.3 クエリタイムアウトパラメータ追加 ● 5.7 ユーザ検索に skip, limit パラメータ追加 ● 415 Unsupported Media Type エラー記述追加
6.0.0	2017/2/21	● グループ名の規定について追記(UTF-8 を許容、先頭 "_EXT-" は予約) ● 7.バケット管理にインデックス取得・シャードキー取得 API を追加

		● 5.1 サインアップ ● 5.4 ユーザ情報更新 ● 5.6 パスワードリセット要求 ➢ 本人確認/通知メール設定有効時の記載を追記 ● 10.1 SSE Push 受信の認証方式を追記 ● 11. サーバ情報取得 API 追加
6.2.0	2017/4/27	● ローカル/外部認証連携ユーザ混在を許可 ➢ User に federated フィールドを追加 ➢ 5.1 サインアップ の注意事項を変更 ● 11.2 ヘルスチェック API 追加 ● 5.2 ログイン ➢ ワンタイムトークンでのログイン方式を追記 ➢ LDAP 認証時のグループ情報の扱いについて追記 ● 5.10 SAML 認証開始 ➢ 新規 API を作成 ● 4.4 429 Too Many Requests エラーを追加
6.5.0	2017/9/27	● クライアント証明書認証対応 ● OpenID Connect 認証対応 ➢ User に primaryLinkedUserId フィールドを追加 ➢ 5.11 OpenID Connect 認証開始 API 追加 ➢ 5.12 リンク情報取得 API 追加 ➢ 5.13 リンク情報削除 API 追加 ➢ 5.1 サインアップの注意事項に追記 ➢ 5.2 ログインの補足・注意事項に追記 ● ファイルストレージ ➢ ユーザ定義メタ情報の付与 ● グループ管理 ➢ グループ名に使用できる文字から '/' を除外
7.0.0	2017/9/28	● 11.3. API 統計情報取得 ➢ 新規 API を作成
7.0.1	2017/10/19	● 4.4.1 ステータスコード 503 の要因にライセンス違反を追加 ● 7.1 バケットの作成・更新のレスポンスコード 403 にバケット数制限超過を追加 ● 以下 API のレスポンスコードに 413 Request Entity Too Large ステータスコードを追加 ➢ 8.1. オブジェクトの作成（Create） ➢ 8.5. オブジェクトの更新 ➢ 9.1. ファイルの新規アップロード ➢ 9.2. ファイルの更新アップロード ● 8.8 バッチオペレーションの実行結果に requestEntityTooLarge を追加
7.0.2	2017/12/22	● 8.9. オブジェクトの集計(Aggregation) API を追加

目次

1. はじめに

本文書は、モバイルバックエンド基盤 REST API のリファレンスである。

本文書では各機能毎の REST API の具体的な仕様について定める。

2. 表記について

2.1. APIパスの表記

モバイルバックエンド基盤 REST API エンドポイント URI は以下のように定義する。

```
https://API サーバホスト名/api/バージョン番号/テナント ID/サービス種別
```

- API サーバホスト名、および "/api/" までのパスは、SI/システム毎に異なる。
- バージョン番号は、本 REST API 仕様では "1" 固定。
- サービス種別は、REST API の種別を識別する文字列。

以下 API の説明時には、上記のうちバージョン番号より後ろのパスのみを表記する。

2.2. ACL

ACL は、グループ、オブジェクトストレージ(バケット、オブジェクト)、ファイルストレージ(バケット、ファイル)などに付与し、アクセス制限を行うためのものである。ACL には、対象データのオーナ、読み書き可能なユーザ・グループ、管理ユーザ・グループのリストを記述する。

ACL の表記は基本的に共通のため、ここで表記方法について説明する。

2.2.1. ACLの基本表記

ACL は JSON で以下のように記述する。

```
{
    "owner": "514af36644f9cb2eb8000001",
    "r" : [ "g:authenticated" ],
    "w" : [ "514af36644f9cb2eb8000002", "514af36644f9cb2eb8000003", "g:group1" ],
    "c" : [ "g:group2" ],
    "u" : [ "g:group2" ],
    "d" : [],
    "admin": [ "514af36644f9cb2eb8000004", "g:group3" ],
}
```

ACL は、以下の情報から構成される。

- "owner": 対象のオーナとなるユーザ ID
  - owner は以下 r, w, c, u, d, admin 権限を包含する(バケットに対する ACL の場合は admin 権限のみを包含する)
- "r": 対象に対する read 可能なユーザ ID/グループ名の一覧
- "w": 対象に対する write 可能なユーザ ID/グループ名の一覧。"w" は以下の属性を包含する。
  - "c": 対象に対する create (追加)可能なユーザ ID/グループ名の一覧 (contentACL のみ)
  - "u": 対象に対する update (更新)可能なユーザ ID/グループ名の一覧
  - "d" 対象に対する delete (削除)可能なユーザ ID/グループ名の一覧
- "admin": 本 ACL を変更可能なユーザ ID/グループ名の一覧

ACL 内でグループを記述するときは、グループ名の先頭に "g:" プレフィクスを付与する。また、以下の特殊なグループ名を定義する。

- "authenticated"： ログイン認証された全ユーザを表す
- "anonymous"： ログインしていないユーザを含む全ユーザを表す

2.2.2. データに対する ACL

ACL は JSON 形式で表記し、対象の JSON 内に埋め込む。以下にオブジェクトストレージのオブジェクトデータに埋め込む場合の例を示す。以下のように、ACL は "ACL" キー内に埋め込む。
ただし、オーナとなるユーザが設定されていない場合は、owner フィールドは存在しない。

```
{
    "_id": "xxxxxxx",
    "itemName": "Computer 12345",
    "price": 120000,
    "date": "2014-03-04",
    "ACL": {
        "owner": "514af36644f9cb2eb8000002",
        "r": [ "g:authenticated" ],
        "w": [ "514af36644f9cb2eb8000002", "514af36644f9cb2eb8000003" ],
        "u": [],
        "d": [],
        "admin" : [ "514af36644f9cb2eb8000003" ],
    }
}
```

2.2.3. バケットに対する ACL 表記

また、対象がバケットの場合は、バケットそのものに対する ACL と、バケットの中身に対する contentACL の2つを埋め込む形となる。
contentACL には owner および admin フィールドは存在しない。また c フィールドが追加されている。

```
{
    "name": "BucketName",
    "ACL": {
        "owner": "514af36644f9cb2eb8000002",
        "r": [ "g:authenticated" ],
        "w": [],
        "u": [],
        "d": [],
        "admin": [ "514af36644f9cb2eb8000003" ],
    },
    "contentACL": {
        "r": [ "g:authenticated" ],
        "w": [ "514af36644f9cb2eb8000002", "514af36644f9cb2eb8000003" ],
        "c": [],
        "u": [],
        "d": []
    }
```

```
}
```

ACL に関するその他表記
本仕様書では、以下のような記述を用いる。

- read 権限が必要 ⇒ ACL の "r" で許可されていること。
- create 権限が必要 ⇒ ACL の "w" または "c" で許可されていること。
- update 権限が必要 ⇒ ACL の "w" または "u" で許可されていること。
- delete 権限が必要 ⇒ ACL の "w" または "d" で許可されていること。
- admin 権限が必要 ⇒ ACL の "admin" で許可されていること。

なお対象がバケット以外（オブジェクトやファイルなど）の場合、対象の owner には上記権限は自動的に付与される。また対象がバケットの場合、対象の owner には admin 権限が自動的に付与される。
このような場合、owner 権限については明記しない。

2.3. 仮想バケット名

ユーザ管理、グループ管理については、以下の2つの仮想バケットがある。

- "_USERS"： ユーザの読み取り・追加・変更・削除に関するバケットコンテンツ ACL を保持する。
- "_GROUPS"： グループの読み取り・追加・変更・削除に関するバケットコンテンツ ACL を保持する。

バケット管理については、以下の仮想バケットがある。

- "_ROOT"： バケットの追加に関するバケットコンテンツ ACL を保持する。

2.4. その他の表記

'{' と '}' で囲まれた表記は、それぞれ以下のように読み替えること。

- {tenant_id}： テナント ID。テナント毎に割り当てられるユニークな ID。
  - テナント ID は、テナント作成時に割り当てられ、デベロッパーコンソール上で確認できる。
  - テナント ID は、MongoDB の ObjectID の文字列表記である。これは 16 進数表記 24 文字の文字列である。
  - 例) "514af36644f9cb2eb8000002"
- {app_id}： アプリケーション ID。アプリケーション毎に割り当てられるユニークな ID。
  - アプリケーション ID は、アプリケーション作成時に割り当てられ、デベロッパーコンソール上で確認できる。
  - テナント ID 同様、MongoDB の ObjectID の文字列表記である。
- {app_key}： アプリケーションキー。アプリケーション毎に割り当てられる秘密キー。アプリケーション ID 同様、アプリケーション作成時に割り当てられ、デベロッパーコンソール上で確認できる。
- {bucket_name}： バケット名
  - バケット名の詳細はバケット管理の章を参照。
- {user_name}： ユーザ名
  - ユーザ名の詳細はユーザ管理（ユーザ操作）の章を参照。
- {group_name}： グループ名
  - グループ名の詳細はユーザ管理（グループ管理系）の章を参照。
  - 以下のグループ名は予約語のため使用できない。
    - ✧ authenticated
    - ✧ anonymous

3. 認証

モバイルバックエンド基盤 REST API では、以下2種類のクライアント認証を実施する。

- アプリケーション認証
- ユーザ認証

3.1. アプリケーション認証

「アプリケーション認証」は、モバイルバックエンド基盤サーバが個々のアプリケーションを認証するものである。

モバイルバックエンド基盤を利用するすべてのアプリケーションには、アプリケーション ID とアプリケーションキーの2つが発行される。この2つはデベロッパコンソール上で確認できる。

API 実行時には、この2つを HTTP ヘッダ(後述)に含めることで、アプリケーションを認証する。アプリケーション ID/アプリケーションキーは、通常クライアントアプリケーション内に埋め込む形となる。

3.2. ユーザ認証

「ユーザ認証」は、モバイルバックエンド基盤アプリケーションを利用する個々のユーザを認証するものである。

ユーザは通常 ID とパスワードをサーバに提示し認証を行う。認証が完了すると、モバイルバックエンド基盤サーバはセッショントークンをクライアントに払い出す。クライアントは以後の API 呼び出しでセッショントークンを提示することで、API 認証を行う。セッショントークンには、HTTP ヘッダ(後述)を使用する。

3.3. クライアント証明書認証

「クライアント証明書認証」は、上記ユーザ認証と同様にモバイルバックエンド基盤アプリケーションを利用する個々のユーザを認証するものである。

「ユーザ認証」では ID とパスワードを提示するのに対し、「クライアント証明書認証」では、認証局(CA)から発行される証明書を用いてユーザを認証する。ID とパスワードが不要になるため、従来のサインアップ／ログイン処理無しで該当するユーザの権限で API を呼び出すことができる。

ただし、クライアント証明書認証を利用するためには、AP サーバの前段に HAProxy サーバの設置が必須となる。

クライアント証明書認証利用時は、各 API の「X-Session-Token」ヘッダが不要となる。その代り HAProxy での認証情報は以下のヘッダに格納して、モバイルバックエンド基盤の各 API へ渡されるよう HAproxy 側で設定を行う必要がある。

ヘッダ名	値	説明
X-SSL-Client-CertAuth-Validated	0 … 無効 1 … 有効	クライアント証明書認証が行われた場合に 1 を付与する。 本ヘッダが無い場合、またはその他の値の場合は、0 と同様とみなす。
X-SSL-Client-Serial	文字列	クライアント証明書のシリアルナンバー ユーザ名の要素として使用可能。
X-SSL-Client-CN	文字列	クライアント証明書の Subject に設定された CN(Common Name)を設定する。 ユーザ名の要素として使用可能。
X-SSL-Client-UID	文字列	クライアント証明書の Subject に設定された UID 要素を設定する。 ユーザ名の要素として使用可能。 デバイスの製造番号などの固有値を割り当てること。
X-SSL-Issuer-DN	文字列	クライアント証明書の Issuer フィールドの値。

		テナントの設定に Issuer が登録されていない場合、比較は行われず正常とみなす。
X-SSL-Validate-Token	文字列	モバイルバックエンド基盤のシステム設定値と一致する文字列を指定する。

4. 共通定義

4.1. JSON

REST API は、多くの場面で JSON を使用する。

JSON の記法については http://json.org/json-ja.html を参照のこと。

JSON の value には、JSON の仕様通り、以下のものが指定できる。

- string
- number
- object
- array
- true
- false
- null

4.2. JSONプロパティ名

JSON のプロパティ名は、原則としてすべて camelCase とする(先頭文字を小文字、単語区切りを大文字。アンダースコアは使用しない)。なお、以下のプロパティ名は予約語とし、別の目的では使用できないものとする。

- "_id"：オブジェクトの ID
- "ACL"：アクセス制御リスト
- "contentACL"：バケットコンテンツ ACL
- "createdAt"：オブジェクトの生成時刻
- "updatedAt"：オブジェクトの更新時刻
- "etag": ETag 値
- 先頭がアンダースコア（"_"）、およびハイフン("-") で始まる文字列

4.3. 日付の表記

JSON は日付型を表現する能力がないため、日付は ISO 8601 日付形式の文字列として、以下のように表現する。

```
YYYY-MM-DDThh:mm:ss.sssZ
```

タイムゾーンは UTC 固定とする。秒は小数点以下3桁に固定する。

以下に例を示す。

```
2014-03-12T09:12:53.789Z
```

上記日付表記は、作成日時(createdAt)や更新日時(updatedAt)などで使用される。

なお、オブジェクトストレージにおいて、これ以外のフォーマットでユーザ定義の日時文字列をユーザ定義フィールドに格納することは自由であり、特に制限はない。

4.4. エラー通知

4.4.1. ステータスコード

エラー時に返却するステータスコードは、各 API の「レスポンスコード」に記載しているが、これ以外に以下のステータスコードが返る場合がある。

- 429 Too Many Requests: API レートリミット制限を超過した場合に発生
- 503 Service Unavailable: データベースや MQ に対するアクセスエラーが発生した場合、有効なライセンスキーが設定されていない場合などに発生
- 504 Gateway Timeout: データベースや MQ 接続がタイムアウトした場合などに発生
- 500 Internal Server Error: その他の内部エラー

4.4.2. レスポンスボディ

レスポンスが JSON 形式である REST API においてエラーが発生した場合は、以下の JSON フォーマットでエラーが通知される。

```
{
    "error": "error messages..."
}
```

ただし、各 API の説明において別のエラーレスポンスが定義されている場合はこの限りではない。

4.5. HTTP ヘッダ

4.5.1. アプリケーションID/アプリケーションキー

すべての API 呼び出しにおいて、以下2つのヘッダは必須である。各 API 説明の「HTTP ヘッダ」には特別な場合を除きこれらヘッダは記載しないが、常に送信が必要である。

- X-Application-Id： アプリケーション ID
- X-Application-Key: アプリケーションキー

4.5.2. セッショントークン

以下ヘッダは、セッショントークンを送信するために使用する。

- X-Session-Token： セッショントークン

セッショントークンは、ログイン API 発行時にサーバからクライアントに発行され、ログアウト時に破棄される。
API 発行時の X-Session-Token の送信ルールは以下の通りである。

- クライアントがセッショントークンを保持している場合、X-Session-Token を常時送信する。
- クライアントがセッショントークンを保持していない場合、X-Session-Token は送信しない。

なお、以下 API の説明で X-Session-Token が「必須」となっている場合、セッショントークンを送信せずに API を呼び出すとエラーを返却する（X-Session-Token を付けずに API を呼び出してはならない、という意味ではない）。
また、X-Session-Token が記載されていない場合、サーバは X-Session-Token を参照しない（送信してはならない、という意味ではない）

4.6. クエリパラメータ

各 API に記載している「リクエストパラメータ」は、クエリパラメータとして URL に指定すること。

また、各 API では特に明記していないが、クエリパラメータはすべて適切に URL エンコードすること。

5. ユーザ管理 (ユーザ操作)

5.1. サインアップ

説明	ユーザを登録する
メソッド	POST
パス	/1/{tenant_id}/users
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● Content-Type: "application/json"
リクエストパラメータ	なし
リクエストボディ	JSON 形式で記述されたユーザ情報。 ● username : ユーザネーム(オプション) ● email: E-mail アドレス(必須) ● password : パスワード (必須) ● options: オプション情報(オプション) ● clientCertUser: クライアント証明書ユーザフラグ(boolean, オプション) options フィールドには、JSON 形式で記述された任意のオブジェクトを指定できる。
レスポンスコード	● 200 OK : 正常登録した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 415 Unsupported Media Type: Content-Type 不正 ● 409 Conflict : 既に同一ユーザ(同じ username もしくは email)が存在している。
レスポンス	作成したユーザ情報を含む JSON データ。_id に作成したユーザの ID が格納される。 ● _id :ユーザ ID ● username : ユーザネーム ● email : E-mail アドレス ● options: オプション情報 ● createdAt : ユーザ作成日時 ● updatedAt : ユーザ更新日時 etag : 新規作成・更新の度に変更される固有値 ● etag : 新規作成・更新の度に変更される固有値 ● federated : 外部認証連携有無 (本 API では常に false) ● primaryLinkedUserId : OpenID Connect 認証でユーザ自動生成時のリンクユーザ ID (本 API では常に null) ● clientCertUser : クライアント証明書認証ユーザフラグ(boolean) 本人確認メール送信設定が有効の場合は、空のユーザ情報を返す。
注意事項	● ユーザはアプリ毎ではなく、テナント毎に作成される。 ● サインアップ成功した状態ではまだログインはしていない。改めてログイン API でログインしなおす必要がある。 ● _USERS バケット の contentACL に対する anonymous ユーザの create 権限が必要。 ● username を指定しない場合は、ランダムな半角英数字 8 文字が設定される。 ● username、email、password のポリシは以下とする。 ポリシに違反する場合は、400 エラーを返却する。 ➢ username: 1 文字以上、100 文字以下の 1 バイト文字列 ➢ email: 100 文字以下、メールアドレスとして正しい文字列 ➢ password: 8 文字以上、100 文字以下の 1 バイト文字列

	● 外部認証を行う場合は、以下 API を使用すること。 ➢ LDAP 認証: ログイン API ➢ SAML 認証: SAML 認証開始 API、ログイン API ➢ OpenID Connect 認証: OpenID Connect 認証開始 API、ログイン API ● 本人確認メール設定が有効の場合、本 API 実行時にユーザ登録はされない。API 実行後に、リクエストボディに指定した E-mail アドレス宛に送られる本人確認メールの内容に従い登録を行うこと。 ● 本人通知メール設定が有効の場合、登録後に登録した E-mail アドレス宛に、本人通知メールが送信される。 ● クライアント証明書認証ユーザを登録する場合は clientCertUser に true を指定すること。この場合、username は必須、email, password は不要である。

ユーザを作成する。リクエストボディの例を以下に示す。

```
{
    "username": "tarou",
    "email": "nichiden.tarou@example.com",
    "password": "Passw0rd"
    "options": {
        "displayName": "日電 太郎",
        "division": "日電事業部"
    }
}
```

レスポンスボディの例を以下に示す。

```
{
    "_id": "52116f01ac521e1742000001",
    "username": "foo",
    "email": "foo@example.com",
    "options": {
        "displayName": "日電 太郎",
        "division": "日電事業部"
    },
    "createdAt": "2013-08-27T04:37:30.000Z",
    "updatedAt": "2013-08-27T04:37:30.000Z",
    "etag": "8c92c97e-01a7-11e4-9598-53792c688d1b",
    "federated": false
}
```

5.2. ログイン

説明	ログインを行う。ログインにはユーザ名、E-mail アドレス、ワンタイムトークンのいずれかを用いる。ログインが成功すると、セッショントークンが払い出される。
メソッド	POST
パス	/1/{tenant_id}/login
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● Content-Type: "application/json"

リクエストパラメータ	なし
リクエストボディ	JSON で記述されたログイン情報。 ● username： ユーザネーム（username / email / token の何れか必須） ● email：E-mail アドレス（username / email / token の何れか必須） ● token： ワンタイムトークン(username / email / token の何れか必須) ● password： パスワード(token が指定されていない場合は必須)
レスポンスコード	● 200 OK： 正常登録した ● 400 Bad Request： パラメータ/リクエストボディ不正 ● 401 Unauthorized： 認証エラー ● 415 Unsupported Media Type: Content-Type 不正
レスポンス	ログイン情報とユーザ情報を含む JSON データ。 ● _id :ユーザ ID ● sessionToken： セッショントークン ● expire： セッショントークンの有効期限。UNIX Time (1970-01-01 00:00:00 UTC からの経過秒数)で表現する。デフォルトはログインから 24 時間後。 ● username： ユーザネーム ● email：E-mail アドレス ● groups: ユーザが所属する全グループの一覧（authenticated, anonymous は含まず） ● options: オプション情報 ● createdAt： ユーザ作成日時 ● updatedAt： ユーザ更新日時 ● lastLoginAt： 前回のログイン日時 ● etag： 新規作成・更新の度に変更される固有値。ログイン前後で変更はない。 ● federated: 外部認証連携有無(boolean) ● primaryLinkedUserId：OpenID Connect 認証でユーザ自動生成時のリンクユーザ ID ● clientCertUser： クライアント証明書認証ユーザフラグ(本 API では常に false)
補足・注意事項	● username と email の両方が設定された場合、username のみが有効となり、email は無効となる。 ● LDAP 認証を使う場合は、username と password のみを指定すること。(email を指定しても無視される) ● SAML 認証・OpenID Connect 認証を使う場合は、token のみを指定すること。(他のパラメータを同時に指定しても無視される) ● token には 5.10. SAML 認証開始および 5.11. OpenID Connect 認証開始の呼び出し後に通知されるワンタイムトークンを指定すること。 ● 有効期限の切れた token を指定した場合は認証エラーとなる。 ● SAML 認証・OpenID Connect 認証未使用テナントの場合は、token を指定しても無視される。 ● 以後、ユーザ認証が必要な API では、セッショントークンを X-Session-Token ヘッダに指定すること。 ● セッショントークンには有効期限があるので注意すること。期限切れとなった場合は再度ログインを行うこと。 ● セッショントークンの有効時間はデベロッパーコンソールで指定する。 ● セッショントークンの利用が終わったら、原則ログアウト API を呼び出してトークンを無効化すること。 ● _USERS バケットの contentACL に対する権限は不要。

リクエストボディの例を以下に示す。

ユーザネームとパスワードでログインする場合

```
{
    "username": "tarou",
    "password": "Passw0rd"
}
```

ワンタイムトークンでログインする場合

```
{
    "token": "ujgBHPgmNLDkUkjTapDiHipPzdHiEidKDiaiJHqP"
}
```

レスポンスの例を以下に示す。

```
{
  "_id":"52116f01ac521e1742000001",
  "sessionToken":"eyJ1c2VyX2lkIjp7IiRvaWQiOiAiNTIxMTZmMDFhYzUyMWUxNzQyMDAwMDAxIn0sImV4c
  GlyZSI6MTM3Njg3ODIzOX0=--5d44532510f4ad2236ec39a403dde3c3f704c5cd",
  "expire":1376878239,
  "username": "foo",
  "email": foo@example.com,
  "groups": [ "group1", "group2", "group3" ],
  "options": {
    "displayName": "日電 太郎",
    "division": "日電事業部"
  }
  "createdAt": "2013-08-27T04:37:30.000Z",
  "updatedAt": "2013-08-27T04:37:30.000Z",
  "lastLoginAt": "2013-08-27T04:37:30.000Z "
  "etag": "8c92c97e-01a7-11e4-9598-53792c688d1b",
  "federated": false,
  "primaryLinkedUserId": "5953a6b10b1fed0f61c49ead"
}
```

LDAP 認証を利用した場合は、LDAP サーバを使用して認証に成功後、MongoDB にユーザを登録する。
これにより、各 API が該当ユーザを使用可能となる。
MongoDB に登録するユーザ情報は、下記の通り。

ユーザ情報	説明
id	MongoDB 保存時に自動付加される ID
tenantId	テナント ID
username	LDAP エントリのログイン属性の値 デフォルト: ・ActiveDirectory: “sAMAccountName” の値 ・OpenLDAP: “uid” の値
createdAt	MongoDB 上のユーザ作成日時
updatedAt	MongoDB 上のユーザ更新日時
etag	ログイン時に自動付加される ETAG 値
lastLoginAt	最終ログイン日時
federated	外部認証連携有無(常に true)

LDAP 認証に成功した場合、認証されたユーザに関連する公開グループ情報について、MongoDB への反映を行う。登録、更新対象となるグループ情報は、以下の通り。

グループ情報	説明
id	MongoDB 保存時に自動付加される ID
tenantId	テナント ID
name	LDAP エントリの Group 属性の値
users	グループに所属するユーザー覧
groups	グループに所属する子グループ一覧
createdAt	MongoDB 上のグループ作成日時
updatedAt	MongoDB 上のグループ更新日時
etag	情報変更時に自動付加される ETAG 値
acl	グループ ACL

“acl” には、読み出し権限に "g:authenticated"のみが設定される。

5.3. ログアウト

説明	ログアウトを行う。払いだされたセッショントークンを無効化する。
メソッド	DELETE
パス	/1/{tenant_id}/login
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token: セッショントークン (必須)
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常にログアウトした ● 401 Unauthorized : 認証エラー
レスポンス	ログイン情報を含む JSON データ。 ● _id :ユーザ ID
補足・注意事項	● _USERS バケットの contentACL に対する権限は不要。

レスポンスの例を以下に示す。

```
{
  "_id":"52116f01ac521e1742000001"
}
```

5.4. ユーザ情報変更 (パスワード変更を含む)

説明	パスワード変更を含む、ユーザ情報を変更する
メソッド	PUT
パス	/1/{tenant_id}/users/{user_id}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token: セッショントークン (必須、ただしマスターキー使用時は不要) ● Content-Type: "application/json"
リクエストパラメータ	● etag : ETag 値 (オプション)

リクエストボディ	JSON 形式で記述されたユーザ情報。 username, email, password, options を指定（すべてオプション。指定しなかったものは変更されない） options フィールドには、JSON 形式で記述された任意のオブジェクトを指定できる。
レスポンスコード	● 200 OK： 正常登録した ● 400 Bad Request： パラメータ/リクエストボディ不正 ● 401 Unauthorized： 認証エラー ● 403 Forbidden :権限エラー ● 404 Not Found： 該当ユーザが存在しない ● 409 Conflict：username, E-mail アドレスが衝突している、ETag 不一致（etag_mismatch）、更新が衝突した(request_conflicted) ● 415 Unsupported Media Type: Content-Type 不正
レスポンス	変更したユーザ情報を含む JSON データ。 ● _id :ユーザ ID ● username： ユーザネーム ● email：E-mail アドレス ● groups: ユーザが所属する全グループの一覧（authenticated, anonymous は含まず） ● options: オプション情報 ● createdAt： ユーザ作成日時 ● updatedAt： ユーザ更新日時 ● lastLoginAt： 最終ログイン日時。マスターキー使用時のみ返却する。 ● etag： 新規作成・更新の度に変更される固有値 ● federated: 外部認証連携有無(boolean) ● primaryLinkedUserId：OpenID Connect 認証でユーザ自動生成時のリンクユーザ ID ● clientCertUser： クライアント証明書認証ユーザフラグ(boolean) 更新が成功した場合、etag フィールドに、更新された ETag 値が格納される(データの内容に変更がない場合であっても、常に updatedAt と etag 値は更新される)。 本人確認メール送信設定が有効の場合は、空のユーザ情報を返す。
注意事項	● 変更が可能なのは自ユーザのみである。ただし、マスターキーを使用した場合どのユーザの情報でも変更可能。 ● _USERS バケットの contentACL に対する権限は不要。 ● username、email、password のポリシはサインアップと同様。 ● Conflict の場合はその原因を JSON 形式で返す ● 外部認証(LDAP 等)連携ユーザに対しては、本機能は使用できない。使用した場合は 403 エラーを返却する。 ● クライアント証明書ユーザに対しては、E メールとパスワードの更新は無視される。 ● 本人確認メール設定が有効の場合、本 API 実行時にユーザ更新はされない。API 実行後に、更新前の E-mail アドレス宛に送られる本人確認メールの内容に従い更新を行うこと。 ● 本人通知メール設定が有効の場合、更新後、更新後ユーザの E-mail アドレス宛に、本人通知メールが送信される。E-mail アドレスを更新した場合は、更新前ユーザのアドレスにも送信される。

リクエストボディの例を以下に示す。

```
{
```

```
    "username": "tarou",
    "email": "nichiden.tarou@example.com",
    "password": "Passw0rd",
    "options": {
      "displayName": "日電 太郎",
      "division": "日電事業部"
    }
}
```

Conflict の場合は例えば以下のデータが返却される

```
{
  "reasonCode": "etag_mismatch"
  "detail": {
    "_id": "52116f01ac521e1742000001",
    "username": "foo",
    "email": "foo@example.com",
    "options": {
      "displayName": "日電 太郎",
      "division": "日電事業部"
    },
    "createdAt": "2013-08-27T04:37:30.000Z",
    "updatedAt": "2013-08-27T04:37:30.000Z",
    "lastLoginAt": "2013-08-27T04:37:30.000Z ", (マスターキー使用時のみ返却)
    "etag": "8c92c97e-01a7-11e4-9598-53792c688d1b"
  }
}
```

- reasonCode: 原因を示すコード
  - duplicate_key : データ重複。バケットに設定したインデックスのユニーク制約による。
  - etag_mismatch : 更新・削除処理で ETag が不一致
  - request_conflicted : 更新・削除処理で衝突
- detail: エラーに関するデータ
  - 各 reasonCode に対して、下記内容を返却する。

reasonCode	detail
duplicate_key	“Duplicate Key” (文字列)
etag_mismatch	サーバ側のユーザ情報を含む JSON データ (JSON 形式)
request_conflicted	“Updating conflicted” (文字列)

5.5. ユーザ削除

説明	ユーザを削除する
メソッド	DELETE
パス	/1/{tenant_id}/users/{user_id}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token: セッショントークン (必須、ただしマスターキー使用時は不要)
リクエストパラメータ	● etag : ETag 値 (オプション)

リクエストボディ	なし
レスポンスコード	● 200 OK： 正常に削除した ● 401 Unauthorized： 認証エラー ● 403 Forbidden :権限エラー ● 404 Not Found： 該当ユーザが存在しない ● 409 Conflict：ETag 不一致(etag_mismatch)、削除が衝突した(request_conflicted)
レスポンス	なし
注意事項	● 削除が可能なのは自ユーザのみである。ただし、マスターキーを使用した場合どのユーザでも削除可能。 ● _USERS バケットの contentACL に対する権限は不要。 ● 所属するグループからも削除される。 ● Conflict の場合はその原因を JSON 形式で返す。 ● 外部認証(LDAP 等)連携ユーザに対しては、本機能は使用できない。使用した場合は 403 エラーを返却する。

Conflict の場合は以下のデータが返却される。

- reasonCode： 原因を示すコード
  - etag_mismatch ： 更新・削除処理で ETag が不一致
  - request_conflicted： 更新・削除処理で衝突
- detail： エラーに関するデータ
  - reasonCode に対して、下記内容を返却する。

reasonCode	detail
etag_mismatch	サーバ側のユーザ情報を含む JSON データ (JSON 形式)
request_conflicted	“Deleting conflicted” (文字列)

5.6. パスワードリセット要求

説明	パスワードのリセットを行う。
メソッド	POST
パス	/1/{tenant_id}/request_password_reset
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● Content-Type: "application/json"
リクエストパラメータ	なし
リクエストボディ	● JSON 形式で記述されたユーザ情報。username, email (username / email どちらか必須)を指定。
レスポンスコード	● 200 OK： パスワードリセットメールを正常に送信した ● 401 Unauthorized： 認証エラー ● 400 Bad Request： パラメータ/リクエストボディ不正 ● 403 Forbidden： リクエストリミットを超えた ● 404 Not Found： 該当ユーザが存在しない ● 409 Conflict： 更新が衝突した(request_conflicted) ● 415 Unsupported Media Type: Content-Type 不正
レスポンス	なし
注意事項	● リセットは攻撃に使われることを考慮して時間内にリクエストできる回数の制限を設ける。(10 分に一回) ● _USERS バケットの contentACL に対する権限は不要。

	● 外部認証(LDAP 等)連携ユーザ、及びクライアント証明書ユーザに対しては、本機能は使用できない。使用した場合は 403 エラーを返却する。 ● 本人通知メール設定が有効の場合、パスワード更新後、更新したユーザの E-mail アドレス宛に、本人通知メールが送信される。

パスワードのリセットを行うリクエストボディの例を以下に示す。

```
{
    "username": "tarou",
    "email": "nichiden.tarou@example.com",
}
```

Conflict の場合は以下のデータが返却される。

- reasonCode: 原因を示すコード
  - request_conflicted： 更新・削除処理で衝突
- detail: エラーに関するデータ
  - reasonCode に対して、下記内容を返却する。

reasonCode	detail
request_conflicted	“Updating conflicted” (文字列)

5.7. ユーザ情報の検索

説明	ユーザ情報を検索する
メソッド	GET
パス	/1/{tenant_id}/users
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token： セッショントークン (オプション)
リクエストパラメータ	● username： ユーザ名(オプション) ● email：E-mail アドレス(オプション) ● skip： スキップカウント (オプション) ● limit： 検索数上限。0 は無制限。デフォルト値は 100 件。(オプション)
リクエストボディ	なし
レスポンスコード	● 200 OK： 正常取得した ● 401 Unauthorized： 認証エラー ● 403 Forbidden： 権限エラー ● 404 Not Found： 該当ユーザが存在しない
レスポンス	条件に一致するユーザ情報を含む JSON データ。 ● _id： ユーザ ID ● username： ユーザネーム ● email：E-mail アドレス ● options: オプション情報 ● createdAt： ユーザ作成日時 ● updatedAt： ユーザ更新日時 ● lastLoginAt： 最終ログイン日時。マスターキー使用時のみ返却する。 ● etag： 新規作成・更新の度に変更される固有値 ● federated: 外部認証連携有無(boolean) ● primaryLinkedUserId：OpenID Connect 認証でユーザ自動生成時のリンクユーザ ID

	● clientCertUser：クライアント証明書認証ユーザフラグ(boolean)
注意事項	● _USERS バケットの contentACL に対する read 権限が必要である。 ● レスポンスに、”groups” は含まない。 ● 外部認証(LDAP 等)を使用している場合、MongoDB 上に存在するユーザのみを検索する。つまり、一度もログインしていないユーザは検索できない。

username または email をキーにユーザを検索する。

検索条件を指定しなかった場合は全件検索となる。全件検索時は skip と limit で検索範囲を指定できる。
全件検索時のみレスポンスの "count" に総ユーザ数が返却される。ソート順序は createdAt 昇順固定。

レスポンスの例を以下に示す。

```
{
  "results": [
    {
        "_id":"52116f01ac521e1742000001",
        "username": "foo",
        "email": foo@example.com,
        "options": {
          "displayName": "日電 太郎",
          "division": "日電事業部"
        },
        "createdAt": "2013-08-27T04:37:30.000Z",
        "updatedAt": "2013-08-27T04:37:30.000Z",
        "lastLoginAt": "2013-08-27T04:37:30.000Z ",(マスターキー使用時のみ返却)
        "etag": "8c92c97e-01a7-11e4-9598-53792c688d1b",
        "federated": false,
        "primaryLinkedUserId": "5953a6b10b1fed0f61c49ead"
    },
    {
        ....
    }
  ],
  "count": 500
}
```

5.8. ユーザ情報の取得

説明	ユーザ情報を取得する
メソッド	GET
パス	/1/{tenant_id}/users/{user_id}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token：セッショントークン (オプション)
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK：正常取得した ● 401 Unauthorized：認証エラー

	● 403 Forbidden：権限エラー ● 404 Not Found：該当ユーザが存在しない
レスポンス	指定したユーザ情報を含む JSON データ。 ● _id：ユーザ ID ● username：ユーザネーム ● email：E-mail アドレス ● groups：ユーザが所属する全グループの一覧（authenticated, anonymous は含まず） ● options：オプション情報 ● createdAt：ユーザ作成日時 ● updatedAt：ユーザ更新日時 ● lastLoginAt：最終ログイン日時。マスターキー使用時のみ返却する。 ● etag：新規作成・更新の度に変更される固有値 ● federated：外部認証連携有無(boolean) ● primaryLinkedUserId：OpenID Connect 認証でユーザ自動生成時のリンクユーザ ID ● clientCertUser：クライアント証明書認証ユーザフラグ(boolean)
注意事項	● _USERS バケットの contentACL 対する read 権限が必要である。

レスポンスの例を以下に示す。

```
{
    "_id":"52116f01ac521e1742000001",
    "username": "foo",
    "email": foo@example.com,
    "groups": [ "group1", "group2", "group3" ],
    "options": {
        "displayName": "日電 太郎",
        "division": "日電事業部"
    },
    "createdAt": "2013-08-27T04:37:30.000Z",
    "updatedAt": "2013-08-27T04:37:30.000Z",
    "lastLoginAt": "2013-08-27T04:37:30.000Z ",(マスターキー使用時のみ返却)
    "etag": "8c92c97e-01a7-11e4-9598-53792c688d1b",
    "federated": false,
    "primaryLinkedUserId": "5953a6b10b1fed0f61c49ead"
}
```

5.9. 自ユーザ情報の取得

説明	ログイン中の自ユーザ情報を取得する
メソッド	GET
パス	/1/{tenant_id}/users/current
HTTP ヘッダ	● X-Application-Id：アプリケーション ID（必須） ● X-Application-Key：アプリケーションキー（必須） ● X-Session-Token：セッショントークン（必須）
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK：正常取得した ● 401 Unauthorized：認証エラー

レスポンス	ログイン中の自ユーザ情報を含む JSON データ。 ● _id：ユーザ ID ● username：ユーザネーム ● email：E-mail アドレス ● groups: ユーザが所属する全グループの一覧（authenticated, anonymous は含まず） ● options: オプション情報 ● createdAt：ユーザ作成日時 ● updatedAt：ユーザ更新日時 ● lastLoginAt：最終ログイン日時 ● etag：新規作成・更新の度に変更される固有値 ● federated: 外部認証連携有無(boolean) ● primaryLinkedUserId：OpenID Connect 認証でユーザ自動生成時のリンクユーザ ID ● clientCertUser：クライアント証明書認証ユーザフラグ(boolean)
注意事項	_USERS バケットの contentACL に対する権限は不要。

レスポンスの例を以下に示す。

```
{
    "_id":"52116f01ac521e1742000001",
    "username": "foo",
    "email": "foo@example.com ",
    "groups": [ "group1", "group2", "group3" ],
    "options": {
      "displayName": "日電 太郎",
      "division": "日電事業部"
    },
    "createdAt": "2013-08-27T04:37:30.000Z",
    "updatedAt": "2013-08-27T04:37:30.000Z",
    "lastLoginAt": "2013-08-27T04:37:30.000Z ",
    "etag": "8c92c97e-01a7-11e4-9598-53792c688d1b",
    "federated": false,
    "primaryLinkedUserId": "5953a6b10b1fed0f61c49ead"
}
```

5.10. SAML認証開始

説明	SAML 認証を開始する
メソッド	GET
パス	/1/{tenant_id}/auth/saml/init
HTTP ヘッダ	なし
リクエストパラメータ	● redirect：認証結果通知先 URL(必須)
リクエストボディ	なし
レスポンスコード	● 302 Found：リダイレクト応答(正常処理) ● 400 Bad Request：リクエストパラメータ不正(指定なし含む) ● 403 Forbidden：SAML 認証未使用テナント
レスポンスヘッダ	● Location：認証サーバの認証 URI ● Set-Cookie：SAML 認証結果受付用 Cookie 情報
レスポンスボディ	エラー情報を含む HTML データ。レスポンスコードが「302 Found」の場合はなし。

注意事項	● 本 API は、SAML 認証画面への誘導と認証成功後のログインに使用するワンタイムトークンの通知を行う(通知方法は後述)。 ● SAML 認証を使用しているテナントに対してのみ実行可能。 ● “redirect” には、デベロッパーコンソールで登録した URL を設定すること。登録されていない場合は 400 エラーを返却する。

レスポンスヘッダの例を以下に示す。

```
Set-Cookie: redirect= http://sample.net;
path=/api/1/52116f01ac521e1742000001/auth/saml/landing; expires=Mon, 31 Jan 2011
14:15:13 GMT
Location:https://baas.example.com/api/1/52116f01ac521e1742000001/auth/saml/start?
idp=http://adfs.example.jp/adfs/services/trust
```

レスポンスボディの例を以下に示す。

```
<html>
  <head>
    <title>400 Error</title>
  </head>
  <body>
    <h1>400 Bad Request</h1>
  </body>
</html>
```

ワンタイムトークンは HTTP リクエストのリクエストパラメータとして認証結果通知先 URL に通知される。リクエストパラメータ名は「token」であり、HTTP リクエストは以下のフォーマットとなる。
　{認証結果通知先 URI}?token={ワンタイムトークン}
HTTP リクエスト行の例を以下に示す。

```
GET http://sample.net?token=t568PrrpvB1W7DhqBhlVfTMAc2NhsIw9j60OTZLz HTTP/1.1
```

SAML 認証に成功し、認証されたユーザと同名のユーザが存在しない場合、MongoDB にユーザを登録する。MongoDB に登録するユーザ情報は、下記の通り。

ユーザ情報	説明
id	MongoDB 保存時に自動付加される ID
tenantId	テナント ID
username	SAML トークンの CommonName 属性の値
createdAt	MongoDB 上のユーザ作成日時
updatedAt	MongoDB 上のユーザ更新日時
etag	ログイン時に自動付加される ETAG 値
federated	外部認証連携有無(常に true)

“username”について、SAML トークン内に CommonName がない場合は、NameID を用いる。

SAML 認証に成功した場合、認証されたユーザに関連する公開グループ情報について、MongoDB への反映を行う。グループ情報は、グループ階層を意識しないフラット構造として扱う。登録、更新対象となるグループ情報は、以下の通り。

グループ情報	説明
id	MongoDB 保存時に自動付加される ID
tenantId	テナント ID
name	SAML トークンの Group 属性の値
users	グループに所属するユーザー覧
groups	グループに所属する子グループ一覧
createdAt	MongoDB 上のグループ作成日時
updatedAt	MongoDB 上のグループ更新日時
etag	情報変更時に自動付加される ETAG 値
acl	グループ ACL

“acl” には、読み出し権限に "g:authenticated"のみが設定される。

5.11. OpenID Connect認証開始

説明	OpenID Connect 認証を開始する
メソッド	GET
パス	/1/{tenant_id}/auth/oidc/start
HTTP ヘッダ	なし
リクエストパラメータ	● sessionToken： セッショントークン(オプション) ● redirect： 認証結果通知先 URL(必須) ● op：OP 種別(必須) ● scope：Authentication Request に使用する scope 値(オプション) ● createUser： ユーザの自動生成許可フラグ。デフォルト false(オプション)
リクエストボディ	なし
レスポンスコード	● 302 Found： リダイレクト応答(正常処理) ● 400 Bad Request： リクエストパラメータ不正(指定なし含む) ● 401 Unauthorized： 認証エラー ● 403 Forbidden：OpenID Connect 認証未使用テナント
レスポンスヘッダ	● Location： 認証サーバの認証 URI
レスポンスボディ	エラー情報を含む HTML データ。レスポンスコードが「302 Found」の場合はなし。
注意事項	● 本 API は、OpenID Connect 認証画面への誘導と認証成功後のログインに使用するワンタイムトークンの通知を行う(通知方法は後述)。 ● OpenID Connect 認証を使用しているテナントに対してのみ実行可能。 ● “redirect” には、デベロッパーコンソールで登録した URL を設定すること。登録されていない場合は 400 エラーを返却する。

sessionToken を省略した場合は、新規ログイン向けの OpenID Connect 認証処理を開始する。

既に存在するユーザに OpenID Connect 認証された OP アカウントをリンクする場合は、リンクさせたいユーザでログイン後、sessionToken を設定して本 API を実行する。リンク設定した OP アカウントでの OpenID Connect 認証ログインにより、リンク設定したユーザでログインすることができる。1 ユーザに対して複数の OP アカウントをリンク設定できる。
sessionToken を設定した場合、認証に使用された OP アカウントが既に自ユーザでリンク設定されている場合は、正常処理となる。別ユーザでリンク設定されている場合は、リンクエラーを通知する。

op に設定可能な値を以下に示す。

google	Google 認証
adfs	ADFS (Windows Server 2016)
other	OpenAM を使用

scope で認証時に取得するユーザ情報種別の指定が可能。scope 値はスペース区切りの文字列を設定する。
scope 値を設定する場合は、必ず "openid" が含まれなければならない。
scope 値を省略した場合は、OP で利用可能なユーザ情報を全て取得する。
ユーザ情報の取得が不要の場合は、"openid" のみを設定する。

レスポンスヘッダの例を以下に示す。

```
Location: https://accounts.google.com/o/oauth2/v2/auth?client_id=0123456789-
abcde.apps.googleusercontent.com&redirect_uri=http://baas.example.com/api/1//591c215bd6
f6880f652109d1/auth/oidc/auth_resp&response_type=code&scope=openid%20email%20profile
```

レスポンスボディの例を以下に示す。

```
<html>
  <head>
    <title>400 Error</title>
  </head>
  <body>
    <h1>400 Bad Request</h1>
  </body>
</html>
```

ワンタイムトークンは HTTP リクエストのリクエストパラメータとして認証結果通知先 URL に通知される。
認証・リンク成功時には、リクエストパラメータ「token」にワンタイムトークンが設定される。
認証・リンクエラー時は、リクエストパラメータ「error」にエラー理由が設定される。
HTTP リクエストは以下のフォーマットとなる。

- 認証・リンク成功 : {認証結果通知先 URI}?token={ワンタイムトークン}
- 認証・リンクエラー : {認証結果通知先 URI}?error={エラー理由}

HTTP リクエスト行の例を以下に示す。

```
GET http://sample.net?token=t568PrrpvB1W7DhqBhlVfTMAc2NhsIw9j60OTZLz HTTP/1.1
```

OpenID Connect 認証に成功し、認証されたユーザと同名のユーザが存在しない場合、MongoDB にユーザを登録する。このとき createUser が false の場合は、ユーザを登録せずに認証エラーを通知する。
MongoDB に登録するユーザ情報は、下記の通り。

ユーザ情報	説明
id	MongoDB 保存時に自動付加される ID
tenantId	テナント ID
options	JSON 配列型の"claims"フィールドに、認証時に取得したユーザ情報が JSON 文字列で格納される
createdAt	MongoDB 上のユーザ作成日時
updatedAt	MongoDB 上のユーザ更新日時
etag	ログイン時に自動付加される ETAG 値
federated	外部認証連携有無(常に true)
primaryLinkedUserId	リンクユーザ ID 設定されたリンクユーザ ID は、リンク情報削除不可となる

“username”, “email”にはランダムな文字列が設定される。
options の claims フィールドには、認証時に OP から取得した ID Token と UserInfo を足し合わせた claim セットが JSON 文字列で設定される。claims フィールドは JSON 配列型であり、リンクされている全ての claim セットが設定される。具体的な claim 名は、OpenID Connect Core 1.0 を参照。options は、ユーザ登録時だけでなく、OpenID Connect 認証成功時に常に更新される。

OpenID Connect 認証に成功時には MongoDB にリンクユーザの登録・更新を行う。
MongoDB に登録するリンクユーザ情報は、下記の通り。

リンクユーザ情報	説明
id	MongoDB 保存時に自動付加される ID
userId	リンクされたユーザ ID
iss	Issuer
sub	Subject
op	OP 種別

5.12. OpenID Connectリンク情報取得

説明	ユーザに設定されたリンク情報を取得する
メソッド	GET
パス	/1/{tenant_id}/users/{user_id}/links
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token: セッショントークン (必須、ただしマスターキー使用時は不要)
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK： 正常取得した ● 401 Unauthorized： 認証エラー ● 403 Forbidden： 権限エラー、OpenID Connect 認証未使用テナント ● 404 Not Found： 該当ユーザが存在しない
レスポンス	リンク情報の配列を含む JSON データ。
注意事項	● 取得が可能なのは自ユーザのみである。ただし、マスターキーを使用した場合どのユーザでも取得可能。

レスポンスの例を以下に示す。

```
{
  "results": [
    {
        "_id": "5953a6b10b1fed0f61c49ead",
        "iss": "https://accounts.google.com",
        "sub": "20930060227221555063 1",
        "op" : "google"
    },
    {
        ....
    }
  ]
}
```

5.13. OpenID Connectリンク情報削除

説明	ユーザに設定されたリンク情報を削除する
メソッド	DELETE
パス	/1/{tenant_id}/users/{user_id}/links/{linkedUserId}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token: セッショントークン (必須、ただしマスターキー使用時は不要)
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK： 正常に削除した ● 401 Unauthorized： 認証エラー ● 403 Forbidden： 権限エラー、OpenID Connect 認証未使用テナント ● 404 Not Found： 該当リンクが存在しない。該当ユーザがリンク情報と一致しない
レスポンス	なし
注意事項	● 削除が可能なのは自ユーザのみである。ただし、マスターキーを使用した場合どのユーザでも削除可能。 ● 対象のリンク情報が primaryLinkedUserId に設定されている場合は 403 エラーを返却する

リンク情報削除に成功した場合は、該当ユーザの以下の情報を更新する。

- options: 該当リンクの claim 情報を削除
- federated: 全てのリンクが削除された場合に false を設定

6. ユーザ管理 (グループ管理系)

6.1. グループの作成

説明	グループを作成する。
メソッド	POST
パス	/1/{tenant_id}/groups/{group_name}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション) ● Content-Type: "application/json"
リクエストパラメータ	なし
リクエストボディ	JSON 形式で記述されたグループ情報。 ● users : ユーザー覧 (オプション) ● groups : グループ一覧 (オプション) ● ACL : ACL (オプション)
レスポンスコード	● 200 OK : 正常登録した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 409 Conflict : 同じグループ名の重複登録 ● 415 Unsupported Media Type: Content-Type 不正
レスポンス	作成したグループ情報を含む JSON データ。
注意事項	● 本 API は、HTTP の POST メソッドを利用し、グループを新規作成する。 ● グループはアプリ毎ではなく、テナント毎に作成される。 ● グループ作成時は_GROUPS バケットの contentACL に対する create 権限が必要。 ● ”users”と”groups”には、既に登録済みのユーザとグループを指定すること。未登録のユーザ・グループを指定した場合は 400 エラーが返却される。 ● “users”と”groups”を指定しない場合は、下記とする。 ➢ グループ作成時: users, groups を空で作成 ➢ グループ更新時: users, groups を更新しない

グループ名は group_name に指定する。グループ名には 100 文字以下の '/' を除く任意の UTF-8 文字列が使用できる。ただし、先頭が "_EXT-" で開始するグループ名は特殊用途に予約されているため使用できない。

リクエストボディで、グループの定義を記述する。以下に例を示す。

```
{
    "users": [ "xxxxx", "yyyyy", "zzzzz" ],
    "groups" : [ "group2", "group3" ],
    ACL: {
        // ACL
    }
}
```

- users に、ユーザ ID の一覧を指定する。
- groups に、このグループが包含する他のグループ名の一覧を指定する。
- ACL には、このグループの読み込み・変更を設定する。

上記例では、作成したグループは group2, group3 を包含するため、group2 および group3 に所属するユーザ、および group2 や group3 が包含する他のグループのユーザも、本グループに所属するとみなされ

る。
また、自分が所属するグループを確認したい場合は、「6.3. グループ一覧の取得」、「6.4. グループの取得」、「5.9. 自ユーザ情報の取得」のいずれかの API を実行する。

ACL を指定しなかった場合は以下の ACL が自動的に設定される。

- ログイン済みの場合: 全フィールドが空。オーナ(作成ユーザ)のみがアクセス可。
- 未ログイン状態の場合 : r, w に "g:anonymous" が設定され、誰でも読み書き可。

6.2. グループの変更

説明	グループを変更する。該当グループが存在しないとき、グループ作成となる。
メソッド	PUT
パス	/1/{tenant_id}/groups/{group_name}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション) ● Content-Type: "application/json"
リクエストパラメータ	● etag : ETag 値 (オプション)
リクエストボディ	JSON 形式で記述されたグループ情報。 ● users : ユーザー一覧 (オプション) ● groups : グループ一覧 (オプション) ● ACL : ACL (オプション)
レスポンスコード	● 200 OK : 正常登録した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 409 Conflict : ETag 不一致(etag_mismatch)、更新が衝突した(request_conflicted) ● 415 Unsupported Media Type: Content-Type 不正
レスポンス	作成したグループ情報を含む JSON データ。 更新が成功した場合、etag フィールドに、更新された ETag 値が格納される(データの内容に変更がない場合であっても、常に updatedAt と etag 値は更新される)。
注意事項	● グループはアプリ毎ではなく、テナント毎に作成される。 ● グループ作成時は_GROUPS バケットの contentACL に対する create 権限が必要。 ● グループ変更時は_GROUPS バケットの contentACL に対する update 権限、および対象グループの update 権限の両方が必要。ACL の変更を伴う場合は、対象グループの admin 権限も必要。 ● グループ変更時は_GROUPS バケットの contentACL、および対象グループの read 権限が無くても、上記のレスポンスが返却される。 ● ”users”と”groups”には、既に登録済みのユーザとグループを指定すること。未登録のユーザ・グループを指定した場合は 400 エラーが返却される。 ● “users”と”groups”を指定しない場合は、下記とする。 ➢ グループ作成時: users, groups を空で作成 ➢ グループ更新時: users, groups を更新しない ● Conflict の場合はその原因を JSON 形式で返す

グループ名は group_name に指定する。グループ名には 100 文字以下の '/' を除く任意の UTF-8 文字列が使用できる。ただし、先頭が "_EXT-" で開始するグループ名は特殊用途に予約されているため使用できない。

リクエストボディで、グループの定義を記述する。以下に例を示す。

```
{
    "users": [ "xxxxx", "yyyyy", "zzzzz" ],
    "groups" : [ "group2", "group3" ],
    ACL: {
        // ACL
    }
}
```

子グループの包含とACLの指定に関する補足事項は「6.2 グループの変更」と同様。そちらへ参照。

Conflict の場合は例えば以下のデータが返却される

```
{
  "reasonCode": "etag_mismatch"
  "detail": {
    "_id":"52116f01ac521e1742000001",
    "name": "group3",
    "users": [ "xxxxx", "yyyyy", "zzzzz" ],
    "groups" : [ "group2", "group3" ],
    "ACL": {…},
    "createdAt": "2013-08-27T04:37:30.000Z",
    "updatedAt": "2013-08-27T04:37:30.000Z"
    "etag": "8c92c97e-01a7-11e4-9598-53792c688d1b"
  }
}
```

- reasonCode: 原因を示すコード
  - etag_mismatch : 更新・削除処理で ETag が不一致
  - request_conflicted : 更新・削除処理で衝突
- detail: エラーに関するデータ
  - reasonCode に対して、下記内容を返却する。

reasonCode	detail
etag_mismatch	サーバ側のグループ情報を含む JSON データ (JSON 形式)
request_conflicted	“Updating conflicted” (文字列)

6.3. グループ一覧の取得

説明	グループ一覧を取得する
メソッド	GET
パス	/1/{tenant_id}/groups
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常取得した ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー

レスポンス	グループ情報の配列を含む JSON データ。
注意事項	● _GROUPS バケットの contentACL および対象グループの read 権限が必要。

レスポンスの例を以下に示す。

```
{
  "results": [
    {
        "_id":"52116f01ac521e1742000001",
        "name": "group3",
        "users": [ "xxxxx", "yyyyy", "zzzzz" ],
        "groups" : [ "group2", "group3" ],
        "ACL": {…},
        "createdAt": "2013-08-27T04:37:30.000Z",
        "updatedAt": "2013-08-27T04:37:30.000Z",
        "etag": "8c92c97e-01a7-11e4-9598-53792c688d1b"
    },
    {
        ....
    }
  ]
}
```

6.4. グループの取得

説明	グループの情報を取得する
メソッド	GET
パス	/1/{tenant_id}/groups/{group_name}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常取得した ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 404 Not Found : グループが存在しない。
レスポンス	グループ情報 JSON データ。
注意事項	● _GROUPS バケットの contentACL および対象グループの read 権限が必要。

グループ名は group_name に指定する。グループ名には '/' を除く任意の UTF-8 文字列が使用できる。

レスポンスの例を以下に示す。

```
{
    "_id":"52116f01ac521e1742000001",
    "name": "group3",
    "users": [ "xxxxx", "yyyyy", "zzzzz" ],
    "groups" : [ "group2", "group3" ],
```

```
    "ACL": {…},
    "createdAt": "2013-08-27T04:37:30.000Z",
    "updatedAt": "2013-08-27T04:37:30.000Z",
    "etag": "8c92c97e-01a7-11e4-9598-53792c688d1b"
}
```

6.5. グループ削除

説明	グループを削除する
メソッド	DELETE
パス	/1/{tenant_id}/groups/{group_name}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	etag : ETag 値 (オプション)
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常取得した ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 404 Not Found : グループが存在しない。 ● 409 Conflict : ETag 不一致(etag_mismatch)、削除処理が衝突した(request_conflicted)
レスポンス	なし
注意事項	● _GROUPS バケットの contentACL および対象グループの delete 権限が必要。 ● 所属するグループからも削除される。 ● Conflict の場合はその原因を JSON 形式で返す

グループ名は group_name に指定する。グループ名には 100 文字以下の '/' を除く任意の UTF-8 文字列が使用できる。ただし、先頭が "_EXT-" で開始するグループ名は特殊用途に予約されているため使用できない。

Conflict の場合は以下のデータが返却される。

- reasonCode: 原因を示すコード
  - etag_mismatch : 更新・削除処理で ETag が不一致
  - request_conflicted : 更新・削除処理で衝突
- detail: エラーに関するデータ
  - reasonCode に対して、下記内容を返却する。

reasonCode	detail
etag_mismatch	サーバ側のグループ情報を含む JSON データ (JSON 形式)
request_conflicted	“Deleting conflicted” (文字列)

6.6. 所属ユーザ・グループの追加

説明	グループに、所属するユーザ・グループを追加する
メソッド	PUT
パス	/1/{tenant_id}/groups/{group_name}/addMembers
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須)

	● X-Application-Key: アプリケーションキー（必須） ● X-Session-Token： セッショントークン（オプション） ● Content-Type: "application/json"
リクエストパラメータ	なし
リクエストボディ	グループに追加する、JSON 形式で記述されたユーザ・グループ情報。 ● users： ユーザ ID 一覧（オプション） ● groups： グループ名一覧（オプション）
レスポンスコード	● 200 OK： 追加成功 ● 400 Bad Request： リクエストボディ不正 ● 401 Unauthorized： 認証エラー ● 403 Forbidden： 権限エラー ● 404 Not Found： グループ(“group_name”)が存在しない。 ● 409 Conflicted： 更新が衝突した(request_conflicted) ● 415 Unsupported Media Type: Content-Type 不正
レスポンス	ユーザ・グループを追加後のグループ情報(JSON データ)。 追加に成功した場合、ETag 値が更新される。
注意事項	● _GROUPS バケットの contentACL に対する update 権限、および対象グループの update 権限の両方が必要。 ● GROUPS バケット、および対象グループの read 権限が無くても、上記のレスポンスが返却される。 ● ”users”と”groups”には、既に登録済みのユーザとグループを指定すること。未登録のユーザ・グループを指定した場合は 400 エラーを返却する。 ● 下記の場合は 200 OK を返却する。更新日時と ETAG 値は更新される。 ➢ 既に所属しているユーザ・グループを指定した場合 ➢ “users”と”groups”を共に指定しない場合(= リクエストボディが空欄)

グループ名は group_name に指定する。グループ名には 100 文字以下の '/' を除く任意の UTF-8 文字列が使用できる。ただし、先頭が "_EXT-" で開始するグループ名は特殊用途に予約されているため使用できない。

リクエストボディで、追加するユーザ・グループを記述する。以下に例を示す。

```
{
    "users": [ "xxxxx", "yyyyy", "zzzzz" ],
    "groups" : [ "group2", "group3" ],
}
```

- users に、ユーザ ID の一覧を指定する。
- groups に、グループ名の一覧を指定する。

Conflict の場合は以下のデータが返却される。

- reasonCode： 原因を示すコード
  - ➢ request_conflicted： 更新・削除処理で衝突
- detail： エラーに関するデータ
  - ➢ reasonCode に対して、下記内容を返却する。

reasonCode	detail
request_conflicted	“Updating conflicted” (文字列)

6.7. 所属ユーザ・グループの削除

説明	グループから、所属するユーザ・グループを削除する
メソッド	PUT
パス	/1/{tenant_id}/groups/{group_name}/removeMembers
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション) ● Content-Type: "application/json"
リクエストパラメータ	なし
リクエストボディ	グループから削除する、JSON 形式で記述されたユーザ・グループ情報。 ● users : ユーザ ID 一覧 (オプション) ● groups : グループ名一覧 (オプション)
レスポンスコード	● 200 OK : 削除成功 ● 400 Bad Request : リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 404 Not Found : グループ(“group_name”)が存在しない。 ● 409 Conflicted : 更新が衝突した(request_conflicted) ● 415 Unsupported Media Type: Content-Type 不正
レスポンス	ユーザ・グループを削除後のグループ情報(JSON データ)。 削除に成功した場合、ETag 値が更新される。
注意事項	● _GROUPS バケットの contentACL に対する update 権限、および対象グループの update 権限の両方が必要。 ● GROUPS バケット、および対象グループの read 権限が無くても、上記のレスポンスが返却される。 ● グループに所属していないユーザ・グループが指定された場合はそのユーザ・グループは無視される。 ● 下記の場合は 200 OK を返却する。更新日時と ETAG 値は更新される。 ➢ “users”と”groups”を共に指定しない場合(= リクエストボディが空欄) ● 指定したグループの所属ユーザ・グループから削除されるのみで、DB 上からは削除されない。

グループ名は group_name に指定する。グループ名には 100 文字以下の '/' を除く任意の UTF-8 文字列が使用できる。ただし、先頭が "_EXT-" で開始するグループ名は特殊用途に予約されているため使用できない。

Conflict の場合は以下のデータが返却される。

- reasonCode: 原因を示すコード
  - ➢ request_conflicted : 更新・削除処理で衝突
- detail: エラーに関するデータ
  - ➢ reasonCode に対して、下記内容を返却する。

reasonCode	detail
request_conflicted	“Updating conflicted” (文字列)

7. バケット管理

7.1. バケットの作成・更新

説明	オブジェクトバケットを作成・更新する
メソッド	PUT
パス	/1/{tenant_id}/buckets/{bucket_type}/{bucket_name}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション) ● Content-Type: "application/json"
リクエストパラメータ	なし
リクエストボディ	JSON 形式で記述されたバケット情報。 ● ACL : ACL (作成時:オプション、 更新時:必須) ● contentACL : コンテンツ ACL (作成時:オプション、 更新時:必須) ● description : バケットの説明 (作成時:オプション、 更新時:必須)
レスポンスコード	● 200 OK : 正常登録した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー。バケット数制限超過。 ● 409 Conflict : 同名のバケット作成が衝突。更新と削除処理が衝突。 ● 415 Unsupported Media Type: Content-Type 不正
レスポンス	作成したバケット情報を含む JSON データ。リクエスト時と同じものだが、owner が設定される。
注意事項	● バケットはテナント毎に作成される。 ● 1 テナント当たり作成できるバケットの個数には上限がある(数千個程度)。 ● テナント内では、同一名称のバケットを複数作成することはできない。 ● 通常のバケット名は先頭が英数字、2文字目以降は英数字または'_'でなければならない。 ただし、特殊バケットの更新時は「_ROOT,_USERS,_GROUPS」を指定できる ● バケット名は 40 文字以下でなければならない。 ● バケット作成時には、_ROOT バケットの contentACL に対する create 権限が必要。 ● バケット更新(ACL/contentACL 以外)を行う際は、対象バケットに対する update 権限が必要。 ● バケットの ACL または contentACL を変更する際は、対象バケットに対する admin 権限が必要。 ● 対象バケットに対する read 権限が無くても、上記のレスポンスが返却される。

オブジェクトストレージおよびファイルストレージで使用するためのバケットを作成する。

{bucket_type} にバケットのタイプを指定する。オブジェクトストレージの場合は "object"、ファイルストレージの場合は "file" を指定する。

{bucket_name} にバケットの名前を指定する。

リクエストボディは、バケットの情報を JSON で指定する。以下に例を示す。

```
{
    "description": "バケットの説明文",
    "ACL": {
        "r": [ "g:authenticated" ],
```

```
        "admin": [],
    },
    "contentACL": {
        "r": [ "g:authenticated" ],
        "w": [ "514af36644f9cb2eb8000002" ]
    }
}
```

ここではバケット名 "bucket1" のオブジェクトバケットを作成する。
description には、バケットの説明文を指定することができる。

なお、ACL を指定しなかった場合は以下の ACL が自動的に設定される。

- ACL:
  - ログイン済みの場合: "owner" にログインユーザが設定される。"r" に "g:authenticated" が設定され、認証ユーザは誰でもバケット情報を読み出せる。
  - 未ログイン状態の場合: "r" に "g:anonymous" が設定され、誰でもバケット情報を読み出せる。
- contentACL :
  - ログイン済みの場合: r, w に "g:authenticated" が設定され、認証ユーザは誰でも読み書き可。
  - 未ログイン状態の場合 : r, w に "g:anonymous" が設定され、誰でも読み書き可。

description、ACL、contentACL はバケット作成時は全てオプションであるため、どれも指定せず作成することができる。
その場合のリクエストは以下の通り、どのフィールドも持たない空の JSON となる。

```
{}
```

7.2. バケット一覧取得

説明	オブジェクトバケットの一覧を取得する
メソッド	GET
パス	/1/{tenant_id}/ buckets/{bucket_type}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常に取得した ● 401 Unauthorized : 認証エラー
レスポンス	バケット情報一覧
注意事項	● 対象バケットに対する read 権限が必要。

{bucket_type} にバケットのタイプを指定する。オブジェクトストレージの場合は "object"、ファイルストレージの場合は "file" を指定する。

バケット情報一覧は以下のように JSON 配列で返却される。

```
{
  "results":[
    {"name": "bucket1","description": "説明文 1","ACL":{…},"contentACL":{…}},
    {"name": "bucket2","description": "説明文 2","ACL":{…},"contentACL":{…}},
    {"name": "bucket3","description": "説明文 3","ACL":{…},"contentACL":{…}}
```

```
  ]
}
```

7.3. バケット情報の取得

説明	オブジェクトバケットの情報を取得する
メソッド	GET
パス	/1/{tenant_id}/ buckets/{bucket_type}/{bucket_name}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常に取得した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 404 Not Found : 指定したバケットが存在しない
レスポンス	バケット情報
注意事項	● 対象バケットに対する read 権限が必要である。

{bucket_type} にバケットのタイプを指定する。オブジェクトストレージの場合は "object"、ファイルストレージの場合は "file" を指定する。

レスポンスボディでは、以下のようにバケット名と ACL の情報が返却される。

```
{
    "name": "bucket_name",
    "description": "説明文",
    "ACL": {
        .....
    },
    "contentACL": {
        .....
    }
}
```

7.4. バケットの削除

説明	バケットを削除する
メソッド	DELETE
パス	/1/{tenant_id}/buckets/{bucket_type}/{bucket_name}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常に削除した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー

	● 403 Forbidden： 権限エラー ● 404 Not Found： 指定したバケットが存在しない ● 409 Conflict： データが存在しているため削除できない
レスポンス	なし
注意事項	● バケットを削除する際、事前にバケット内に格納されている全データを削除しておかなければならない。 ● 対象バケットに対する delete 権限が必要。 ● ただし、マスターキーを設定し、バケットを削除する場合は、バケット内にデータの有無によらず、バケットとその中に格納された全データが強制削除される。

7.5. インデックス一覧の取得

説明	オブジェクトバケットのインデックス一覧情報を取得する
メソッド	GET
パス	/1/{tenant_id}/buckets/object/{bucket_name}/index
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token： セッショントークン (オプション)
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK： 正常に取得した ● 400 Bad Request： パラメータ/リクエストボディ不正 ● 401 Unauthorized： 認証エラー ● 403 Forbidden： 権限エラー ● 404 Not Found： 該当バケットが存在しない
レスポンス	インデックス情報
注意事項	● 対象バケットの admin 権限が必要である。

レスポンスでは、バケットに設定されているインデックスの配列が返却される。

```
{ "results": [
    {
        "name": "key1_key2",
        "keys": [
           { "name": "key1", "type": 1 },
           { "name": "key2", "type": -1 }
        ]
    },
    {
        "name": "key3",
        "keys": [
            { "name": "key3", "type": 1 }
        }
    }
] }
```

7.6. シャードキーの取得

説明	オブジェクトバケットのシャードキーを取得する

メソッド	GET
パス	/1/{tenant_id}/buckets/object/{bucket_name}/shardkey
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常に取得した ● 400 Bad Request : パラメータ不正。DB がシャード構成ではない場合。 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 404 Not Found : 対象バケットが存在しない。シャードキー未設定。
レスポンス	シャードキー情報
注意事項	● 対象バケットの admin 権限が必要である。 ● MongoDB がシャード構成でない場合 400 BadRequest となる。 ● シャード構成で、シャードキー未設定の場合は空の JSON が返却される。

シャードキー情報は以下のように JSON で返却される。

```
{
    "key": [
        { "name": "key1", "type": 1 }
    ],
    "unique": false
}
```

この例は、key1 の昇順でシャードキーが設定されており、ユニーク制約は設定されていないことを示す。

8. オブジェクトストレージ

8.1. オブジェクトの作成 (Create)

説明	オブジェクトを作成する
メソッド	POST
パス	/1/{tenant_id}/objects/{bucket_name}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション) ● Content-Type: "application/json"
リクエストパラメータ	● requestToken : リクエストトークン (オプション)
リクエストボディ	JSON 形式で記述されたオブジェクト。詳細は後述。
レスポンスコード	● 200 OK : 正常登録した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden :権限エラー ● 404 Not Found : バケットが存在しない ● 409 Conflict : データ重複エラー(duplicate_key)、ID 衝突(duplicate_id) ● 413 Request Entity Too Large: オブジェクトサイズ制限超過 ● 415 Unsupported Media Type: Content-Type 不正 ● 500 InternalServerError : シャードキーの制約によるエラー。その他。
レスポンス	作成したオブジェクト情報を含む JSON データ。リクエスト時と同じものだが、_id などのフィールドが追加される。 Conflict の場合はその原因を JSON 形式で返す。
注意事項	● バケット contentACL の create 権限が必要 ● MongoDB の制約として次のフィールド名は使用できない。先頭が"$"であるフィールド名。 "."(ピリオド)を含むフィールド名

リクエストボディでは、JSON 形式で記述された任意のオブジェクトを指定できる。

以下に例を示す。

```
{
  "name": "Foo",
  "score": 80,
  "ACL": {
    "r": ["g:authenticated"],
    "w": ["g:authenticated"],
  }
}
```

"ACL" には ACL を指定できる。ACL を指定しなかった場合は以下の ACL が自動的に設定される。

- ログイン済みの場合: 全フィールドが空。オーナ(作成ユーザ)のみがアクセス可。
- 未ログイン状態の場合 : r, w に "g:anonymous" が設定され、誰でも読み書き可。owner フィールドは設定されない。

また、クライアント側で "_id" フィールドを指定することができる。_id を指定した場合、サーバは _id の値をそのまま受け入れるが、同一の _id を持つデータが既に存在する場合は 409 Conflict エラーを返す。_id は MongoDB のオブジェクト ID と同じフォーマットでなければならない。
_id を指定しない場合は、サーバ側で _id を採番する。

また、クライアント側から requestToken を指定することができる。サーバは requestToken が指定された場合、レスポンスをキャッシュする。同一 requestToken を持つリクエストが再送された場合、サーバは処理を行わずキャッシュしたレスポンスのみを返却する。なお、クライアントは requestToken をランダムかつ推測不可能な文字列として作成しなければならない。

結果は以下のように返却される。

```
{
  "_id": "521c36d4ac521e1ffa000007",
  "name": "Foo",
  "score": 80
  "ACL": {
    "owner": "xxxxx",
    "r": ["g:authenticated"],
    "w": ["g:authenticated"],
  },
  "createdAt": "2013-08-27T05:19:16.000Z",
  "updatedAt": "2013-08-27T05:19:16.000Z",
  "etag": "8c92c97e-01a7-11e4-9598-53792c688d1b"
}
```

追加されるフィールドは以下のとおり。これらのフィールドは予約名のため、これ以外の用途に使用してはならない。

- _id：オブジェクト ID
- createdAt：作成日時
- updatedAt：更新日時
- ACL：指定しなかった場合自動追加される。指定した場合でも owner がない場合は owner は自動設定される。
- etag：ETag 値。新規作成・更新の度に変更される固有値。

オブジェクトストレージでは Conflict の場合は以下のデータが返却される

```
{
  "reasonCode": "etag_mismatch"
  "detail": {
    "_id": "521c36d4ac521e1ffa000007",
    "name": "Foo",
    "ACL": {…略…},
    "createdAt": "2013-08-27T05:19:16.000Z",
    "updatedAt": "2013-08-27T05:19:16.000Z",
    "etag": "8c92c97e-01a7-11e4-9598-53792c688d1b"
  }
}
```

- reasonCode: 原因を示すコード
  - request_conflicted : 更新・削除処理が衝突した場合。
  - duplicate_key : データ重複。バケットに設定したインデックスのユニーク制約による。
  - duplicate_id : ID 重複。オブジェクト作成時に指定した ID が既に存在する
  - etag_mismatch : 更新・削除処理で Etag が不一致
- detail: エラーに関するデータ
  - Etag 不一致(etag_mismatch) の場合のみサーバ側のデータが JSON 形式で格納される
  - それ以外場合はエラーメッセージが格納される。

8.2. オブジェクトの読み込み

説明	オブジェクトを取得する
メソッド	GET
パス	/1/{tenant_id}/objects/{bucket_name}/{object_id}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	● deleteMark : 削除マークされたデータを読み込む (オプション)
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常終了 ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden :権限エラー ● 404 Not Found : 該当バケットまたはデータが存在しない
レスポンス	オブジェクト情報を含む JSON データ。
注意事項	● バケット contentACL と対象オブジェクトの read 権限が必要

deleteMark パラメータに 1 を指定した場合、削除マークされたデータも読み込まれる。削除マークされたデータは、以下のように "_deleted" フィールドに true が設定されている。

```
{
  "_id": "521c36d4ac521e1ffa000007",
  "name": "Foo",
  "ACL": {
    "owner": "xxxxx",
    "r": ["g:authenticated"],
    "w": ["g:authenticated"],
  },
  "createdAt": "2013-08-27T05:19:16.000Z",
  "updatedAt": "2013-08-27T05:19:16.000Z",
  "etag": "8c92c97e-01a7-11e4-9598-53792c688d1b",
  "_deleted": true
}
```

8.3. オブジェクトのクエリ

説明	オブジェクトをクエリする
メソッド	GET
パス	/1/{tenant_id}/objects/{bucket_name}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	● where : 検索条件 (オプション) ● order : ソート順序 (オプション) ● skip : スキップカウント (オプション) ● limit : 検索数上限。最大値およびデフォルト値は 100 件。(オプション) ● count : 検索条件に合致する全件数取得 (オプション) ● deleteMark : 削除マークされたデータを読み込む (オプション) ● projection : プロジェクション (オプション) ● readPreference : 参照する DB の指定 (オプション)

	● timeout: クエリタイムアウト（ミリ秒，オプション）
リクエストボディ	なし
レスポンスコード	● 200 OK： 正常終了 ● 400 Bad Request： パラメータ/リクエストボディ不正 ● 401 Unauthorized： 認証エラー ● 403 Forbidden： 権限エラー ● 404 Not Found： バケットが存在しない ● 500 InternalServerError： 検索条件・プロジェクションの演算子の利用方法が正しくない。その他のエラー。 ● 504 Gateway Timeout： クエリタイムアウト
レスポンス	オブジェクト情報を含む JSON データ。
注意事項	● バケット contentACL と対象オブジェクトの read 権限が必要 ● デフォルトの検索上限数は 100 件である ● リクエストパラメータが非常に多い場合、ブラウザによっては URL の長さ制限のためにリクエストが発行できない場合がある。この場合はロングクエリ API を使用すること。 ● 検索条件はリクエストパラメータ/URL に格納され、サーバのアクセスログに記録されるため、機密性の高い情報を格納するべきではない。やむを得ず検索式に含めたい場合は、ロングクエリ API を使用すること。

バケット内のオブジェクトをクエリする。結果は以下のように "results" に配列形式で格納される。
また、"currentTime" にクエリ実行時のサーバ時刻が格納される(このフィールドはデータ同期処理を行う際に使用することを想定している)。

```
{
  "results":[
    {
      "_id":"52117490ac521e5637000001",
      "name":"Foo",
      "score":80,
      "ACL": { /* ACL ... */ }
      "createdAt": "2013-08-27T05:19:16.000Z",
      "updatedAt": "2013-08-27T05:19:16.000Z",
      "etag": "8c92c97e-01a7-11e4-9598-53792c688d1b"
    }
  ],
  "currentTime": "2013-09-01T12:34:56.000Z"
}
```

deleteMark に 1 を指定すると、削除済みデータもクエリ対象となる。

8.3.1. 検索条件 (where)

条件指定は where パラメータで指定する。where パラメータには、JSON で検索条件を設定する。

特定のキーに対して完全一致させたい場合は、以下のように指定する。

```
where={"name": "Foo"}
```

比較を行う場合は、以下の演算子を利用できる。

- $lt, $gt：Less Than / Greater Than

- $lte, $gte：Less or Equal / Greater of Equal
- $ne：Not equal to

```
例）score が 70 超のものを選択したい場合
where={"score": {"$gt": 70}}
```

その他、以下の演算子を使用できる。これらは MongoDB の演算子と同じものがそのまま使用できる。

- $in
- $all
- $regex
- $exists
- $not
- $or
- $and

8.3.2. ソート順序 (order)

順序指定は order パラメータで行う。order パラメータには、ソートを行うキー名を指定する。

以下の例は、age キー昇順でソートする。

```
order=age
```

複数条件を指定するときは "," で区切る。また、降順で検索したい場合は検索キーの前に "-" を付与する。
以下の例は、第 1 優先で年齢昇順、第 2 優先で名前降順でソートする。

```
order=age,-name
```

【注意】ソート対象のキー名の先頭が"-"(ハイフン)または、キー名に","(カンマ)が含まれる場合、正常にソートできない場合がある。

【注意】ソート対象のキーにインデックスが設定されていない場合、MongoDB の制限により、ソート対象のデータ量は 32MB 以内に収める必要がある。この制限を超えると 500 Internal Server Error となる。大量のデータをソートしたい場合は、必ずインデックスを設定すること。

8.3.3. スキップカウント(skip) / 検索数上限 (limit)

skip で検索結果の先頭からのスキップ数、limit で返却する検索数の上限を指定する。

以下の例では、検索結果の 100 件目から 50 件を取得する。

```
skip=100&limit=50
```

limit のデフォルト値は 100 件とする。limit を指定しなかった場合は、デフォルトでこの値が指定されたものとみなされる。

limit に -1 を指定した場合は制限なし(無限大)として扱う。

注: サーバ側のコンフィグレーションによっては、limit 値に制限がかけられている場合がある。この場合、上限値を越える値を指定したり -1 を指定した場合は 400 Bad Request エラーとなる。

8.3.4. 件数取得

条件に合致した全件数を取得したい場合は、count パラメータを 1 にセットする。これは通常、skip/limit と組み合わせて使用する。

```
count=1&limit=0
```

として検索した場合の例を以下に示す。

```
{
  "results":[
  ],
  "count": 539
}
```

limit=0 を指定しているため、results の内容は空となるが、count に件数が取得される。

8.3.5. プロジェクション (projection)

検索結果に含まれるオブジェクトのフィールドを制限したい場合には projection に JSON 形式で指定する。projection は MongoDB と同様の書式で指定する。
特定のフィールドのみを取得したい場合は以下のように指定する。

```
projection={"name":1}
```

上記を指定した場合、以下のような結果が得られる。

```
{
  "results":[
    {
      "_id":"52117490ac521e5637000001",
      "name":"Foo"
    }
  ],
  "currentTime": "2013-09-01T12:34:56.000Z"
}
```

逆に、特定のフィールドのみ抑制したい場合は以下のように指定する

```
projection={"name":0}
```

上記を指定した場合、以下のような結果が得られる。(name のみ省略される)

```
{
  "results":[
    {
      "_id":"52117490ac521e5637000001",
      "score":80,
      "ACL": { /* ACL ... */ }
      "createdAt": "2013-08-27T05:19:16.000Z",
      "updatedAt": "2013-08-27T05:19:16.000Z",
      "etag": "8c92c97e-01a7-11e4-9598-53792c688d1b"
    }
  ],
  "currentTime": "2013-09-01T12:34:56.000Z"
}
```

"_id"フィールドはデフォルトで付加されるが、省きたい場合は以下のように指定する。

```
projection={"name":1, "_id":0}
```

上記を指定した場合、以下のような結果が得られる。

```
{
  "results":[
    {"name":"Foo"}
  ],
  "currentTime": "2013-09-01T12:34:56.000Z"
}
```

入れ子構造になっているフィールドを指定する場合は以下のようにする。

```
projection={"parent.child":1}
```

上記を指定した場合、以下のような結果が得られる。

```
{
  "results":[
    {
      "_id":"52117490ac521e5637000001",
      "parent": {"child": 123}
    }
  ],
  "currentTime": "2013-09-01T12:34:56.000Z"
}
```

フィールドは複数指定する事が可能だが、"0"と"1"を混在させることはできない。但し、「"_id":0」のみ例外的に併用可能。

正しい例

```
projection={"name":1, "age":1}
projection={"name":0, "age":0}
projection={"name":1, "age":1, "_id":0}
```

正しくない例

```
projection={"name":1, "age":0}
projection={"name":0, "age":0, "_id":1}
```

また、array フィールドに対して、MongoDB と同様に以下の演算子を使用することも可能。

- $elemMatch
- $slice

【注意】 "$", "$meta" 演算子は対象外。

8.3.6. readPreference

使用する MongoDB がレプリカセットで構築されている場合のみ有効なパラメータ。
参照する DB を以下の値で指定できる。(大文字小文字は区別されない)
指定しない場合はデフォルトで primary となる。

primary	プライマリから読み込む(デフォルト)
secondaryPreferred	セカンダリ優先で読み込む セカンダリが複数ある場合はランダムに選択される

	セカンダリが存在しない場合はプライマリから読み込む

これ以外の値を指定した場合は BadRequest となる

【注意】
セカンダリから読む場合、古い情報が得られる場合がある。これは、プライマリからセカンダリへの複製処理で若干の遅延が発生するため。最新状態を取得する必要がある場合はセカンダリを指定しないこと。

8.4. オブジェクトのクエリ(ロングクエリ)

説明	オブジェクトをクエリする(ロングクエリ)
メソッド	POST
パス	/1/{tenant_id}/objects/{bucket_name}/_query
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション) ● Content-Type: "application/json"
リクエストパラメータ	なし
リクエストボディ	クエリ条件を格納した JSON オブジェクト
レスポンスコード	クエリ API に加え、以下のコードがある。 ● 415 Unsupported Media Type: Content-Type 不正
レスポンス	クエリ API と同じ
注意事項	クエリ API と同じ。ただしリクエストパラメータのサイズ制限は無い。

バケット内のオブジェクトをクエリする。前述のクエリ API と機能は完全に同じである。違いはメソッドが POST である点と、クエリ条件をリクエストパラメータではなくリクエストボディで指定するためクエリ長の制限がない点のみである。

クエリ条件は JSON で表記する。例を以下に示す。

```
{
    "where": {"score": {"$gt": 70} },
    "order": "age,-name",
    "skip": 500,
    "limit": 100,
    "count": 0,
    "deleteMark": 0,
    "projection": { "name": 1, "age": 0 },
    "readPreference": "primary",
    "timeout": 0
}
```

各プロパティの意味は、クエリ API のリクエストパラメータと同じである。各プロパティはそれぞれ省略が可能。

8.5. オブジェクトの更新

説明	オブジェクトを更新する
メソッド	PUT
パス	/1/{tenant_id}/objects/{bucket_name}/{object_id}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須)

	● X-Session-Token： セッショントークン（オプション） ● Content-Type: "application/json"
リクエストパラメータ	● etag：ETag 値（オプション） ●
リクエストボディ	JSON 形式で記述された更新データ。詳細は後述。
レスポンスコード	● 200 OK： 正常登録した ● 400 Bad Request： パラメータ/リクエストボディ不正 ● 401 Unauthorized： 認証エラー ● 403 Forbidden： 権限エラー ● 404 Not Found: 該当バケットまたはデータが存在しない ● 409 Conflict： 更新が衝突した（request_conflicted）、データ重複（duplicate_key）、ETag 不一致（etag_mismatch） ● 413 Request Entity Too Large: オブジェクトサイズ制限超過 ● 415 Unsupported Media Type: Content-Type 不正 ● 500 InternalServerError： シャードキーの制約によるエラー。その他。
レスポンス	更新したオブジェクト情報を含む JSON データ。 更新が成功した場合、etag フィールドに更新された ETag 値が格納される(データの内容に変更がない場合であっても、常に updatedAt と etag 値は更新される)。 Conflict の場合はその原因を JSON 形式で返す。
注意事項	● バケット contentACL と対象オブジェクトの update 権限が必要 ● 対象オブジェクトの ACL の変更を伴う場合、admin 権限も必要。 ● バケット contentACL、および対象オブジェクトの read 権限が無くても、上記のレスポンスが返却される。

リクエストボディの指定によって、部分更新または全更新が可能。

ETag 値を指定した場合、サーバ側データとの ETag 値が照合される。一致していなかった場合、409 Conflict エラーが返却される。レスポンスには、サーバ側のオブジェクト情報を含む JSON データが返却される。
またサーバ側データが削除済みマークされていた場合でも更新することはできる。

サーバ負荷への影響が大きいため、update 処理では再送処理対応(requestToken)をサポートしない。
（サーバ側で処理済みのオブジェクトに対して再送があった場合、Conflict として処理される）

8.5.1. 部分更新

オブジェクトの一部のみを更新する方法。

以下は、score を 80 に変更する例。 指定していないフィールドは変更されない。

```
{"score": 80}
```

MongoDB の $set 演算子を使用することも可能。

```
{"$set": {"score": 80}}
```

$inc 演算子を使用すると、インクリメントを行うことができる。

```
{"$inc": {"score": 10}}
```

他に、 MongoDB の更新演算子 がすべて使用できる。

なお部分更新した場合、"updatedAt" は現在時刻で自動更新される。
"createdAt" は指定しない場合は元の値が保持される。明示的に指定する場合は変更可能だが、正しい日付形式を指定しなければならない。
"_id" は更新できない。

8.5.2. 完全上書き

オブジェクトを完全に上書きする場合は $full_update 演算子を指定する。
ACL は省略不可。

```
{"$full_update": {"name":"Bar", "score":90, "ACL": {"owner": "xxxxx",
"r":["authenticated"], "w":["authenticated"]} }}
```

"updatedAt" は現在時刻に自動更新される。

"createdAt" を指定しなかった場合は、変更前の値が維持される。

8.6. オブジェクトの削除

説明	オブジェクトを削除する
メソッド	DELETE
パス	/1/{tenant_id}/objects/{bucket_name}/{object_id}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	● etag : ETag 値 (オプション) ● deleteMark : 削除マークのみを行う (オプション)
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常終了 ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 404 Not Found: 該当バケットまたはデータが存在しない ● 409 Conflict : 更新処理との衝突(request_conflicted)、ETag 不一致 (etag_mismatch)
レスポンス	完全削除の場合、レスポンスなし(空の JSON) 削除マークの場合、削除マークしたオブジェクト情報を含む JSON データ Conflict の場合はその原因を JSON 形式で返す
注意事項	● バケット contentACL と対象オブジェクトの delete 権限が必要

etag で ETag 値を指定した場合、サーバ側データの ETag 値との比較が行われ、一致しなかった場合は 409 Conflict が返却される。レスポンスには、サーバ側のオブジェクト情報を含む JSON データが返却される。

deleteMark を 1 に指定した場合、実データは削除せず削除マークのみを true に設定する。updatedAt および etag は自動更新される。

```
{
  "_id": "521c36d4ac521e1ffa000007",
  "name": "Foo",     //データの内容は保持される
```

```
  "ACL": {
    "owner": "xxxxx",
    "r": ["g:authenticated"],
    "w": ["g:authenticated"],
  },
  "createdAt": "2013-08-27T05:19:16.000Z",
  "updatedAt": "2013-08-27T05:19:16.000Z",  // 更新される
  "etag": "8c92c97e-01a7-11e4-9598-53792c688d1b", // 更新される
  "_deleted": true // セットされる
}
```

サーバ負荷への影響が大きいため、delete 処理では再送処理対応(requestToken)をサポートしない。
（サーバ側で処理済みのオブジェクトに対して再送があった場合、Conflict として処理される）

8.7. オブジェクトの一括削除

説明	条件に合致するオブジェクトを一括削除する
メソッド	DELETE
パス	/1/{tenant_id}/objects/{bucket_name}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token：セッショントークン（オプション）
リクエストパラメータ	● deleteMark：削除マークのみを行う（オプション） ● where：削除条件（オプション、オブジェクトクエリの検索条件と同様）
リクエストボディ	なし
レスポンスコード	● 200 OK：正常終了 ● 400 Bad Request：パラメータ/リクエストボディ不正 ● 401 Unauthorized：認証エラー ● 403 Forbidden：権限エラー ● 404 Not Found: 該当バケットが存在しない
レスポンス	削除したオブジェクトの個数を含む JSON データ
注意事項	● バケット contentACL と対象オブジェクトの delete 権限が必要

where で条件指定した場合、その条件と ACL を満たすオブジェクトが削除される。
where を指定しない場合、ACL を満たす全てのオブジェクトが削除対象となる。
deleteMark を 1 にした場合、対象のオブジェクトを一括で論理削除状態にする。updatedAt,etag は自動更新される。既に論理削除状態となっているオブジェクトは対象外となる。

レスポンスには、以下のように削除したオブジェクトの個数が JSON 形式で返却される。

```
{
  "result":"ok",
  "deletedObjects":3
}
```

削除対象が見つからない場合のレスポンスは、正常終了で削除個数 0 となる。

8.8. バッチオペレーション

説明	複数オブジェクトを同時に追加・更新・削除する
メソッド	POST
パス	/1/{tenant_id}/objects/{bucket_name}/_batch
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須)

	● X-Application-Key: アプリケーションキー（必須） ● X-Session-Token：セッショントークン（オプション） ● Content-Type: "application/json"
リクエストパラメータ	● requestToken：リクエストトークン（オプション） ● deleteMark：削除時にマークのみを行う（オプション）
リクエストボディ	JSON 形式で記述されたデータ。
レスポンスコード	● 200 OK：正常登録した ● 400 Bad Request：パラメータ/リクエストボディ不正 ● 415 Unsupported Media Type: Content-Type 不正
レスポンス	結果を含む JSON データ。
注意事項	● バケット contentACL と対象オブジェクトの update 権限が必要 ● 対象オブジェクトの ACL の変更を伴う場合、admin 権限も必要。 ● 同一 requestToken を用いた再送が非常に短い間隔で行われた場合、処理が二重に行われる可能性がある。

複数のデータを同時にバッチ更新する。

クライアント側から requestToken を指定することができる。サーバは、requestToken が指定された場合、レスポンスをキャッシュする。同一 requestToken を持つリクエストが再送された場合、サーバは処理を行わずキャッシュしたレスポンスのみを返却する。ただし、同一 requestToken でのリクエストが非常に短い間隔で再送された場合には、処理が二重に行われる可能性があるため注意が必要。
クライアントは requestToken をランダムかつ推測不可能な文字列として作成しなければならない。

データは以下のように記述する。

```
{
  "requests": [
    {
      "op": "insert"
      "data": {
        "_id": "xxxxxxxx",
        "name": "Foo",
        "score": 80,
        "ACL": {
          "r": ["g:authenticated"],
          "w": ["g:authenticated"],
        }
      }
    },
    {
      "op": "update",
      "_id": "yyyyyyyy",
      "etag": "ff123cc6-01a9-11e4-9be6-d39577f35b51",
      "data": {
        "name": "Foo",
        "score": 80,
        "ACL": {
          "r": ["g:authenticated"],
          "w": ["g:authenticated"],
        },
      }
    },
```

```
    {
      "op": "delete",
      "_id": "zzzzzzzz",
      "etag": "11b6cedc-01aa-11e4-88f1-0b3208ae4242"
    }
  ]
}
```

データは "requests" に配列で指定する。サーバ側ではこの配列の先頭から順に処理が実行される。

各要素には以下の要素を指定する。

- op： 操作。insert, update, delete のいずれかを指定。
- _id： 対象となるオブジェクト ID。update, delete の場合は必須。insert の場合は不要。insert で id 指定する場合は data へ"_id" フィールドを格納すること。
- etag： オブジェクトの ETag。update, delete 操作では、ETag を指定することで楽観ロックが実施される(サーバ側データと ETag 照合がされる)。
- data： 更新データ。insert, update の場合は、それぞれオブジェクトの作成、更新 API で送信するリクエストボディの内容と同一。delete の場合は不要。insert の場合、"_id" を含む場合はその値が採用され、指定しない場合はサーバ側で作成する。
  - 注: update の場合、$full_update を指定しないとフルアップデートにならず部分更新となる。

応答は以下のように返却される。

```
{
  "results": [
    {
      "result": "ok",
      "_id": "...",
      "etag": "...",
      "updatedAt": "...",
      "data": { DATA },
    },
    {
      "result": "conflict",
      " reasonCode ": " etag_mismatch ",
      "_id": "...",
      "etag": "...",
      "updatedAt": "...",
      "data": { DATA },
    },
    {
      "result": "serverError",
      "_id": "..."
    },
    {
      "result": "ok",
      "_id": "...",
      "etag": "...",
      "updatedAt": "...",
      "data": { DATA },
    }
```

```
  ]
}
```

results 配列には、リクエスト時のオペレーションに対応した個数の JSON データが格納される。また、この並び順はリクエストの配列の順番に対応している。データには以下のフィールドが設定される。

- result： 実行結果。以下のいずれか。
  - ok： 正常に操作が完了した。
  - conflict： 衝突が発生した。
  - forbidden：ACL 違反のためアクセス不可
  - notFound： 該当データが存在しない
  - badRequest： リクエスト不正
  - requestEntityTooLarge： オブジェクトサイズ制限超過
  - serverError： その他のエラー
- reasonCode :conflict の理由。Result が conflict の場合のみ付加。以下のいずれか。
  - unspecified： 不明（初期値）
  - request_conflicted： 更新・削除処理衝突（更新・削除中に該当オブジェクトに変更発生）
  - duplicate_key： ユニーク制約エラー（重複エラー）
  - duplicate_id：ID が衝突（重複エラー）
  - etag_mismatch：ETag が衝突（比較エラー）
- _id： オブジェクト ID。insert でエラーとなった場合は付加されない。
- etag： 更新後の ETag 値
- updatedAt： 更新時刻
- data：result が ok の場合、更新されたオブジェクトデータ。作成・更新・削除 API のレスポンスボディと同じもの。エラーの場合は "etag_mismatch" のみデータが格納され、それ以外の場合はなし。

8.9. オブジェクトの集計(Aggregation)

説明	オブジェクトの集計処理(Aggregation)を行う
メソッド	POST
パス	/1/{tenant_id}/objects/{bucket_name}/_aggregate
HTTP ヘッダ	● X-Application-Id: アプリケーション ID（必須） ● X-Application-Key: アプリケーションキー（必須） ● X-Session-Token： セッショントークン（オプション） ● Content-Type: "application/json"
リクエストパラメータ	なし
リクエストボディ	Aggregation 条件を格納した JSON オブジェクト
レスポンスコード	● 200 OK： 正常終了 ● 400 Bad Request： パラメータ/リクエストボディ不正 ● 401 Unauthorized： 認証エラー ● 403 Forbidden： 権限エラー ● 404 Not Found： バケットが存在しない ● 415 Unsupported Media Type: Content-Type 不正 ● 500 InternalServerError：pipeline, options の利用方法が正しくない。その他のエラー。 ● 504 Gateway Timeout： タイムアウト
レスポンス	Aggregation 結果を格納した JSON オブジェクト
注意事項	● バケット contentACL と対象オブジェクトの read 権限が必要 ● $lookup を行う場合は、参照先バケットの contentACL と対象オブジェクトの read 権限が必要

バケット内のオブジェクトの集計(Aggregation)処理を行う。MongoDB の Aggregation と同等の処理である。

Aggregation 条件は JSON で表記しリクエストボディに格納する。フォーマットは以下の通りである。

```
{
    "pipeline": [ ... ],
    "options": { ... }
}
```

- pipline には、MongoDB の aggregation pipeline stages を配列で記述する。
  - 記述方式は MongoDB のマニュアルを参照すること。
  - 使用できるコマンドは以下のものに限定される。
    - $match
    - $lookup
    - $limit
    - $skip
    - $sort
    - $group
    - $bucket
    - $bucketAuto
    - $count
    - $project
    - $addFields
  - $lookup, $addFields を使用する場合は、バックエンドの MongoDB は ver 3.4 以上でなければならない。
- options にはオプションを JSON 形式で指定する。
  - 指定できる値は以下のとおり。意味は MongoDB の aggregate コマンドと同一である。
    - allowDiskUse (boolean)： テンポラリファイルの利用許可
    - maxTimeMS (integer)： 処理時間の上限(ミリ秒)
    - batchSize (integer)： バッチサイズ

なお、ACL によるアクセス制御を行うため、pipeline には以下の修正が行われた上で実施される。ただし、これらの処理はマスターキー使用時には実施されない。

- pipeline の先頭に、該当バケット内ドキュメントに対する ACL 制約を行うための $match が挿入される。
- 各 $lookup の直後に、参照先バケット内ドキュメントに対する ACL 制約を行うための $addFields / $filter が挿入される。

Aggregation 結果は以下のように results に配列で結果が格納された JSON として返却される。

```
{
  "results":[ ... ]
}
```

9. ファイルストレージ

9.1. ファイルの新規アップロード

説明	ファイルを新規アップロードする
メソッド	POST
パス	/1/{tenant_id}/files/{bucket_name}/{filename}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション) ● X-ACL : ACL (オプション) ● X-Meta-Options (オプション) ● Content-Type: ファイルの Content-Type(必須)
リクエストパラメータ	● cacheDisabled : キャッシュ禁止フラグ (オプション)
リクエストボディ	ファイルデータ
レスポンスコード	● 200 OK : 正常登録した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 404 Not Found : バケットが存在しない ● 409 Conflict : ファイル名重複(duplicate_filename)、ファイル(論理削除データ)がロック中(file_locked) ● 413 Request Entity Too Large: ファイルサイズ制限超過 ● 507 Insufficient Storage: ファイルストレージの容量上限オーバー
レスポンス	作成したファイルのメタデータを含む JSON 文字列
注意事項	● ファイルの ACL は X-ACL ヘッダで設定することに注意。 ● 同一名称のファイルがあった場合、409 Conflict エラーとなる。 ● 論理削除データ("_deleted" フィールドが true)と同じファイル名の場合は論理削除データに上書きし、"deleted"フィールドを false にリセットする。 ● バケット contentACL の create 権限が必要。

ACL は X-ACL ヘッダで指定する。ACL は以下の例のように JSON 形式の文字列で指定する。

```
{ "owner":"xxxxx", "r": ["xxx"], "w": ["xxx"], "u": [], "d":[], "admin":[] }
```

- ACL を指定しなかった場合は、以下の ACL が自動的に設定される。指定した場合でも owner がない場合は owner は自動設定される。

- ログイン済みの場合: 全フィールドが空。オーナ(作成ユーザ)のみがアクセス可。
- 未ログイン状態の場合 : r, w に "g:anonymous" が設定され、誰でも読み書き可。

ファイルのメタデータにユーザ定義のメタ情報(options)を含める場合は、X-Meta-Options ヘッダで指定する。X-Meta-Options は以下の例のように JSON 形式の文字列で指定する。

```
{ "owner":"日電 太郎", "fileVersion": "1.0.0" }
```

filename には、UTF-8 文字列を URL エンコードしたものを指定できる。filename の長さは、UTF-8 で 900 バイトを上限とする。なお、以下の文字は使用できない。

- 制御文字 (U+0000～U+001F)
- " (U+0022)

- * (U+002A)
- / (U+002F)
- : (U+003A)
- < (U+003C)
- > (U+003E)
- ? (U+003F)
- バックスラッシュ (U+005C)
- | (U+007C)
- DEL (U+007F)

リクエストパラメータにキャッシュ禁止フラグを指定(cacheDisabled=true)した場合、キャッシュ禁止フラグが ON になる。

結果は以下のように返却される。
ETag はメタデータとファイル本体で別に管理する。

```
    {
      "_id": "533d31c43cbc3dd2d0b32cd1", // メタデータのオブジェクト ID
      "filename": "hello.txt",
      "contentType": "text/plain",
      "length": 12,
      "ACL": { /* ACL */ },
      "createdAt": "2013-08-27T04:37:30.000Z",
      "updatedAt": "2013-08-27T04:37:30.000Z",
      "metaETag": "8c92c97e-01a7-11e4-9598-53792c688d1b",
      "fileETag": "8c92c97e-01a7-11e4-9598-53792c688d1c",
      "cacheDisabled": false, // キャッシュ禁止フラグ（デフォルト false）
      "options": {
          "owner": "日電 太郎",
          "fileVersion": "1.0.0"
      }
}
```

ファイルストレージでは Conflict の場合は以下のデータが返却される

```
{
  "reasonCode": "etag_mismatch"
  "detail": {
      "_id": "533d31c43cbc3dd2d0b32cd1",
      "filename": "hello.txt",
      "contentType": "text/plain",
      "length": 12,
      "ACL": { /* ACL */ },
      "createdAt": "2013-08-27T04:37:30.000Z",
      "updatedAt": "2013-08-27T04:37:30.000Z",
      "metaETag": "8c92c97e-01a7-11e4-9598-53792c688d1b",
      "fileETag": "8c92c97e-01a7-11e4-9598-53792c688d1c",
      "cacheDisabled": false,
      "publicUrl": http://.... ,
      "options": {
          "owner": "日電 太郎",
          "fileVersion": "1.0.0"
      }
  }
```

```
}
```

- reasonCode: 原因を示すコード
  - duplicate_key : データ重複。バケットに設定したインデックスのユニーク制約による。
  - duplicate_filename : ファイル名重複。ファイル作成時に指定したファイル名が存在する。
  - etag_mismatch : 更新・削除処理で ETag が不一致
  - file_locked : ファイルがロック中
- detail: エラーに関するデータ
  - 各 reasonCode に対して、下記内容を返却する。

reasonCode	detail
duplicate_key	“Duplicate Key” (文字列)
duplicate_filename	“Duplicate File Name” (文字列)
etag_mismatch	サーバ側のメタデータ (JSON 形式)
file_locked	“File is locked” (文字列)

9.2. ファイルの更新アップロード

説明	ファイルを更新アップロードする
メソッド	PUT
パス	/1/{tenant_id}/files/{bucket_name}/{filename}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション) ● Content-Type: ファイルの Content-Type(オプション)
リクエストパラメータ	● metaETag : ETag 値 (オプション) ● fileETag : ETag 値 (オプション)
リクエストボディ	ファイルデータ
レスポンスコード	● 200 OK : 正常更新した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 404 Not Found : 該当バケットまたはファイルが存在しない ● 409 Conflict : ファイルがロック中(file_locked)、 ETag 不一致(etag_mismatch) ● 413 Request Entity Too Large: ファイルサイズ制限超過 ● 507 Insufficient Storage: ファイルストレージの容量上限オーバー
レスポンス	更新したファイルのメタデータを含む JSON 文字列。 更新が成功した場合、fileETag フィールドに更新された ETag 値が格納される(データの内容に変更がない場合であっても、常に updatedAt と fileETag 値は更新される)。 Conflict の場合はその原因を JSON 形式で返す。
注意事項	● 本 API では ACL は変更できない。ACL を変更したい場合はメタデータの更新 API を使用すること。 ● 同一名称のファイルがあった場合、上書きされる。書き込み権限がない場合は 403 エラーとなる。 ● バケット contentACL と対象ファイルの update 権限が必要。 ● バケット contentACL、および対象ファイルの read 権限が無くても、上記のレスポンスが返却される。

ETag 値(metaETag, fileETag)を指定した場合、サーバ側ファイルの ETag 値(metaETag, fileETag)が

照合される。一致していなかった場合、409 Conflict エラーが返却される。レスポンスには、サーバ側のメタデータを含む JSON データが返却される。
またサーバ側データが削除済みマークされていた場合でも更新することができ、削除済みマーク(“deleted”フィールド)を false にリセットする。

9.3. ファイルのダウンロード

説明	ファイルをダウンロードする
メソッド	GET
パス	/1/{tenant_id}/files/{bucket_name}/{filename}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常登録した ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 404 Not Found : 該当バケットまたはデータが存在しない ● 409 Conflict : ファイルがロック中(file_locked)
レスポンス	ファイルデータ。 なお、Content-Disposition ヘッダに attachment; filename='*ファイル名*' が設定される。 X-Content-Length ヘッダに、ファイルサイズがバイト数で指定される。 ETag ヘッダ(オプション)に、ファイルの ETag(fileETag)が指定される。 Conflict の場合はその原因を JSON 形式で返す。
注意事項	● バケット contentACL と対象ファイルの read 権限が必要。

レスポンスは、Content-Length ヘッダ付きで返される場合と、Chunked エンコーディング(Content-Length ヘッダなし、Transfer-Encoding: Chunked)で返される場合がある。いずれの場合も、ダウンロードされるファイルサイズは X-Content-Length ヘッダで取得できる。Content-Length ヘッダが指定された場合、 X-Content-Length ヘッダの値は Content-Length ヘッダの値と同一となる。

Chunked エンコーディングを使用中に、サーバ側異常でファイルがダウンロード中に削除された場合、ダウンロードは途中で中断される(エラーにはならない)。この場合は、ダウンロードされたデータサイズと X-Content-Length を比較し、一致していない場合はクライアント側でエラーとすること。

9.4. ファイル一覧の取得

説明	ファイル一覧(メタデータ)をダウンロードする ただし、リクエストパラメータで指定がある場合は、公開済みファイル一覧をダウンロードする
メソッド	GET
パス	/1/{tenant_id}/files/{bucket_name}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	● published:公開済みファイルのメタデーター覧の取得用のフラグ ・フラグが存在、かつ、published=1 である場合、公開済みファイル一覧の取得になる ● deleteMark : 削除マークされたデータを読み込む (オプション)

リクエストボディ	なし
レスポンスコード	● 200 OK：正常登録した ● 400 Bad Request：パラメータ/リクエストボディ不正 ● 401 Unauthorized：認証エラー ● 403 Forbidden：権限エラー ● 404 Not Found：バケットが存在しない
レスポンス	サーバ側の現在時刻と、ファイルメタデータの配列 JSON 文字列
注意事項	● バケット contentACL と対象ファイルの read 権限が必要。

各ファイルのメタデータは JSON 配列で返却される。

```
{
  "currentTime": "2014-11-18T07:34:22:242Z",
  "results": [
    {
      "_id": "533d31c43cbc3dd2d0b32cd1",
      "filename": "hello.txt",
      "contentType": "text/plain",
      "length": 12,
      "ACL": { /* ACL */ },
      "createdAt": "2013-08-27T04:37:30.000Z",
      "updatedAt": "2013-08-27T04:37:30.000Z",
      "metaETag": "8c92c97e-01a7-11e4-9598-53792c688d1b",
      "fileETag": "8c92c97e-01a7-11e4-9598-53792c688d1c",
      "cacheDisabled": false,
      "publicUrl": "http://....",
      "options": {
          "owner": "日電 太郎",
          "fileVersion": "1.0.0"
      }
    }
  ]
}
```

メタデータには以下の情報が含まれる。

- _id: メタデータの ID
- filename：ファイル名
- contentType：Content-Type
- length: ファイルサイズ（バイト数）
- ACL: ACL
- 作成日時
- 更新日時
- metaETag: メタデータの ETag
- fileETag: ファイル本体の ETag
- cacheDisabled: キャッシュ禁止フラグ
- publicUrl：公開 URL（ファイルが公開されている場合）
- _deleted: 削除フラグ(deleteMark パラメータに 1 を指定した場合のみ表示)
- options: ユーザ定義メタ情報(オプション)

公開済みファイル一覧の取得の URL 例）

http://localhost:8080/api/1/533d22cae4b07fe168d2a451/files/textfiles?published=1

deleteMark パラメータに 1 を指定した場合、削除マークされたメタデータも読み込まれる。削除マークされたデータは、以下のように "_deleted" フィールドに true が設定されている。この際、サーバ側でファイル本体は削除されるが、メタデータは削除されずに残る。

9.5. 特定ファイルのメタデータ取得

説明	特定ファイルのメタデータを取得する
メソッド	GET
パス	/1/{tenant_id}/files/{bucket_name}/{filename}/meta
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	● deleteMark : 削除マークされたデータを読み込む (オプション)
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常登録した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 404 Not Found : 該当バケットまたはデータが存在しない
レスポンス	ファイルメタデータ
注意事項	● バケット contentACL と対象ファイルの read 権限が必要。

ファイルメタデータが JSON で得られる。

```
{
    "_id": "533d31c43cbc3dd2d0b32cd1",
    "filename": "hello.txt",
    "contentType": "text/plain",
    "length": 12,
    "ACL": { /* ACL */ },
    "createdAt": "2013-08-27T04:37:30.000Z",
    "updatedAt": "2013-08-27T04:37:30.000Z",
    "metaETag": "8c92c97e-01a7-11e4-9598-53792c688d1b",
    "fileETag": "8c92c97e-01a7-11e4-9598-53792c688d1c",
    "cacheDisabled": false,
    "publicUrl": http://.... ,
    "options": {
        "owner": "日電 太郎",
        "fileVersion": "1.0.0"
    }
}
```

deleteMark パラメータに 1 を指定した場合、削除マークされたメタデータも読み込まれる。削除マークされたデータは、以下のように "_deleted" フィールドに true が設定されている。この際、サーバ側でファイル本体は削除されるが、メタデータは削除されずに残る。

```
{
    "_id": "533d31c43cbc3dd2d0b32cd1",
    "filename": "hello.txt",
    "contentType": "text/plain",
    "length": 12,
    "ACL": { /* ACL */ },
    "createdAt": "2013-08-27T04:37:30.000Z",
    "updatedAt": "2013-08-27T04:37:30.000Z",
    "metaETag": "8c92c97e-01a7-11e4-9598-53792c688d1b",
    "fileETag": "8c92c97e-01a7-11e4-9598-53792c688d1c",
    "cacheDisabled": false,
    "_deleted": true,
    "publicUrl": "http://...." ,
    "options": {
        "owner": "日電 太郎",
        "fileVersion": "1.0.0"
    }
}
```

9.6. ファイルメタデータの更新

説明	特定ファイルのメタデータを更新する
メソッド	PUT
パス	/1/{tenant_id}/files/{bucket_name}/{filename}/meta
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション) ● Content-Type: "application/json"
リクエストパラメータ	● metaETag : ETag 値 (オプション)
リクエストボディ	メタデータ (JSON) ● filename : 更新後のファイル名 ● contentType : コンテンツタイプ ● ACL : ACL ● cacheDisabled : キャッシュ禁止フラグ(true or false) ● options: オプション情報(オプション) options フィールドには、JSON 形式で記述された任意のオブジェクトを指定できる。
レスポンスコード	● 200 OK : 正常登録した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 404 Not Found : 該当バケットまたはファイルが存在しない。 ● 409 Conflict : ファイルがロック中(file_locked)、ETag 不一致 (etag_mismatch)、ファイル名重複(duplicate_filename) ● 415 Unsupported Media Type: Content-Type 不正
レスポンス	更新したメタデータ 更新が成功した場合、metaETag フィールドに更新された ETag 値(metaETag) が格納される(データの内容に変更がない場合であっても、常に updatedAt と metaETag 値は更新される)。 Conflict の場合はその原因を JSON 形式で返す。
注意事項	● バケット contentACL と対象ファイルの update 権限が必要。

	● ACL と cacheDisabled を変更する場合は、admin 権限も必要。 ● バケット contentACL、および対象ファイルの read 権限が無くても、上記のレスポンスが返却される。

更新ができるのは、filename と contentType, ACL , キャッシュ禁止フラグである。length は変更できない。

filename, contentType, ACL , キャッシュ禁止フラグはリクエストボディで指定する。指定したプロパティのみが変更される。

```
{
    "_id": "533d31c43cbc3dd2d0b32cd1",
    "filename": "hello2.txt",
    "contentType": "text/plain",
    "length": 12,
    "ACL": { /* ACL */ },
    "createdAt": "2013-08-27T04:37:30.000Z",
    "updatedAt": "2013-08-27T04:37:30.000Z",
    "metaETag": "8c92c97e-01a7-11e4-9598-53792c688d1b", // 更新される
    "fileETag": "8c92c97e-01a7-11e4-9598-53792c688d1c",
    "cacheDisabled": false,
    "publicUrl": "http://...." ,
    "options": {
        "owner": "日電 太郎",
        "fileVersion": "1.0.0"
    }
}
```

なお、filename に使用できる文字・長さの制限は、新規アップロード時の制限と同じである。

9.7. ファイルの削除

説明	特定ファイルを削除する
メソッド	DELETE
パス	/1/{tenant_id}/files/{bucket_name}/{filename}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	● metaETag : ETag 値 (オプション) ● fileETag : ETag 値 (オプション) ● deleteMark : 削除マークのみを行う (オプション)
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常終了した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 404 Not Found : 該当バケットまたはファイルが存在しない。 ● 409 Conflict : ファイルがロック中(file_locked)、 ETag 不一致(etag_mismatch)
レスポンス	完全削除の場合、レスポンスなし(空の JSON) 削除マークの場合、削除マークしたオブジェクト情報を含む JSON データ Conflict の場合はその原因を JSON 形式で返す

注意事項	● バケット contentACL と対象ファイルの delete 権限が必要。

metaETag と fileETag で ETag 値を指定した場合、サーバ側データの ETag 値との比較が行われ、一致しなかった場合は 409 Conflict が返却される。レスポンスには、サーバ側のメタデータを含む JSON データが返却される。

deleteMark を 1 に指定した場合、実メタデータは削除せず削除マークを true に設定する。updatedAt および metaETag, fileETag は更新される。この際、サーバ側でファイル本体は削除されるが、メタデータは削除されずに残る。以下の例を参照。

```
{
    "_id": "533d31c43cbc3dd2d0b32cd1",
    "filename": "hello2.txt",
    "contentType": "text/plain",
    "length": 12,
    "ACL": { /* ACL */ },
    "createdAt": "2013-08-27T04:37:30.000Z",
    "updatedAt": "2013-08-27T04:37:30.000Z"
    "metaETag": "8c92c97e-01a7-11e4-9598-53792c688d1b",  // 更新される
    "fileETag": "8c92c97e-01a7-11e4-9598-53792c688d1c",  // 更新される
    "cacheDisabled": false,
    "_deleted": true,
    "publicUrl": "http://...." ,
    "options": {
        "owner": "日電 太郎",
        "fileVersion": "1.0.0"
    }
}
```

9.8. ファイルの公開

説明	特定ファイルを公開する
メソッド	PUT
パス	/1/{tenant_id}/files/{bucket_name}/{filename}/publish
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	● metaETag : メタデータの ETag 値 (オプション)
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常終了した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 404 Not Found : 該当バケットまたはファイルが存在しない。 ● 409 Conflict : ファイルがロック中(file_locked)、 ETag 不一致(etag_mismatch)、データ重複(duplicate_key)
レスポンス	公開したファイルのメタデータ 成功した場合、metaETag フィールドに更新された ETag 値(metaETag)が格納

	される(データの内容に変更がない場合であっても、常に updatedAt と metaETag 値は更新される)。 Conflict の場合はその原因を JSON 形式で返す。
注意事項	● すでに公開中のファイルを再公開しても URL は変更されない。 ● バケット contentACL の update 権限が必要。 ● 対象ファイルの admin 権限が必要。 ● バケット contentACL、および対象ファイルの read 権限が無くても、上記のレスポンスが返却される。

ファイルを公開する。公開されたファイルには一意な URL が付与され、一切のアクセス制限なく読み込みアクセスができるので、a タグは img タグの href に指定できるようになる。

レスポンスでは、以下のように publicUrl に公開 URL が格納される。

```
{
  "_id": "533d31c43cbc3dd2d0b32cd1",
  "filename": "hello.txt",
  "contentType": "text/plain",
  "length": 12,
  "ACL": { /* ACL */ },
  "createdAt": "2013-08-27T04:37:30.000Z",
  "updatedAt": "2013-08-27T04:37:30.000Z",
  "metaETag": "8c92c97e-01a7-11e4-9598-53792c688d1b", // 更新される
  "fileETag": "8c92c97e-01a7-11e4-9598-53792c688d1c",
  "cacheDisabled": false,
  "publicUrl": "http://xxxx/yyyyy/hello.txt",
  "options": {
      "owner": "日電 太郎",
      "fileVersion": "1.0.0"
  }
}
```

公開ファイル(publicUrl)のアクセス時には、X-Content-Length 拡張ヘッダにファイルサイズが指定される。

9.9. ファイルの公開解除

説明	特定ファイルの公開を解除する
メソッド	DELETE
パス	/1/{tenant_id}/files/{bucket_name}/{filename}/publish
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	● metaETag : メタデータの ETag 値 (オプション)
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常終了した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 権限エラー ● 404 Not Found : 該当バケットまたはファイルが存在しない。 ● 409 Conflict : ファイルがロック中(file_locked)、 ETag 不一致(etag_mismatch)
レスポンス	ファイルのメタデータ

	成功した場合、metaETag フィールドに更新された ETag 値（metaETag）が格納される(データの内容に変更がない場合であっても、常に updatedAt と metaETag 値は更新される)。 Conflict の場合はその原因を JSON 形式で返す。
注意事項	● バケット contentACL の update 権限が必要。 ● 対象ファイルの admin 権限が必要。 ● バケット contentACL、および対象ファイルの read 権限が無くても、上記のレスポンスが返却される。

10. Push通知

10.1. インスタレーションの登録・再登録

説明	インスタレーションを新規登録・再登録する
メソッド	POST
パス	/1/{tenant_id}/push/installations
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● Content-Type: application/json (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	なし
リクエストボディ	インスタレーション情報 (後述)
レスポンスコード	● 200 OK : 正常登録した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 415 Unsupported Media Type: Content-Type 不正
レスポンス	後述
注意事項	インスタレーション ID は漏洩しないようにすること。漏洩すると、第三者にインスタレーション情報を改ざんされるおそれがある。 同一の deviceToken / pushType ペアがすでに存在していた場合、既存のインスタレーション ID で再登録が行われる。

インスタレーション情報は JSON で以下のように指定する。

```
{
    "_osType": "ios",
    "_osVersion": "10.0.0",
    "_deviceToken": "xxxx",
    "_pushType": "apns",
    "_channels": [ "chan1", "chan2" ],
    "_appVersionCode": 6,
    "_appVersionString": "1.0.5",
    "_allowedSenders": [ "g:anonymous", "55ed50600c69293704e34f9d " ],

    /* ここから下は自由設定 */
    "email": "foo@example.com"
}
```

以下のプロパティは、必須である。

- _osType : OS 種別。以下のいずれか。
  - "ios" : iOS
  - "android" : Android
  - “dotnet” : .NET
  - "java" : Java
  - "js": JavaScript
  - "other": その他/不明
- _osVersion : OS バージョン
  - Android の場合は Build.VERSION.SDK_INT で API レベルを取得する。
  - iOS の場合は UIDevice の systemVersion の値を取得する。
  - .NET、その他不明の場合は ”Unknown” を指定する。

- _deviceToken： デバイス固有トークン。
  - APNs の場合は Device Token を指定すること。16 進数表記(0～9a～f、1 バイト 2 文字分)とすること。
  - FCM / GCM の場合は Registration ID/Token を指定すること。
  - SSE の場合は任意の識別子を使用できる(ANDROID_ID, MAC アドレス, UUID など)
- _pushType： 使用する Push テクノロジを指定する。以下のいずれか。
  - "apns"：APNs
  - "gcm"：FCM/GCM
  - “sse”：SSE Push
  - "sasp"：SASP Push (将来拡張用に予約、本バージョンでは使用不可)
- _channels： 購読するチャネルの一覧。
- _appVersionCode： アプリケーションのバージョンコードを整数値で指定する。
  - Android の場合は AndroidManifest.xml の versionCode を指定する。
  - iOS の場合は CFBundleVersion の値を以下のルールで整数値に変換したものを指定する。
    - x.y.z の場合は、 x * 1,000,000 + y * 1,000 + z (ここで、x/y/z はいずれも整数)
    - x.y の場合は、x * 1,000,000 + y * 1,000
    - x の場合は x * 1,000,000
    - 上記フォーマットに従わない場合は、0 とする。
  - .NET、その他の場合は -1 を整数値で指定する.
- _appVersionString: アプリケーションのバージョンを指定する。
  - Android の場合は AndroidManifest.xml の versionName を指定する。
  - iOS の場合は CFBundleShortVersionString の値を指定する。
  - .NET の場合はアセンブリバージョンの値を指定する
- _allowedSenders: このインスタレーションに対して Push を送信可能なユーザ・グループを配列で指定する。ユーザはユーザ ID で指定し、グループはグループ名の先頭に "g:" を付加して指定する。この設定に関わらず、デベロッパコンソールおよびマスターキーを指定した送信は可能である。

上記以外のプロパティについては、クライアント側で自由に設定して良い。これらは Push 送信時にインスタレーションを絞り込む際に使用できる。

正常登録された場合、以下のようにリクエストと同等の JSON に _id が設定されたレスポンスが返却される。_id はインスタレーション ID で、インスタレーションの更新・削除時に必要になるため、クライアント側で保存しておかなければならない。

また、ログイン状態でインスタレーションを登録した場合、自動的に "owner" フィールドにユーザ ID が設定される。

```
{
    "_id": "541bc0a0e4b03de4b0287b2e",
    "_osType": "ios",
    "_osVersion": "10.0.0",
    "_deviceToken": "xxxx",
    "_pushType": "apns",
    "_channels": [ "chan1", "chan2" ],
    "_appVersionCode": 6,
    "_appVersionString": "1.0.5",
    "email": "foo@example.com",
    "_owner": "yyyyy"
}
```

同一の deviceToken / pushType ペアがすでに存在していた場合、インスタレーション情報を上書きし、更新後のデータが返却される。

10.1.1. SSE Push

SSE Push を使用する場合、レスポンスには以下のように "_sse" キー以下の "username", "password", "uri" プロパティに、ユーザ名とパスワード、Push サーバの URI がそれぞれ設定される。

```
{
    "_id": "541bc0a0e4b03de4b0287b2e",
    /* 中略 */,
    "_sse": {
        "username": "xxx",
        "password": "yyy",
        "uri": "https://push-server.example.com/events/foo/bar"
    }
}
```

上記 uri に接続することで、Server-Sent Event でイベントの受信を行うことができる。
この際、username と password の値を使用して認証を行う必要がある。認証には BASIC 認証を用いる。
HTTPS を使用し、認証情報が漏洩しないようにすること。

10.2. インスタレーション情報の取得

説明	インスタレーション情報を取得する
メソッド	GET
パス	/1/{tenant_id}/push/installations/{installationId}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須)
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常取得 ● 401 Unauthorized : 認証エラー ● 404 NotFound : 該当するインスタレーションが存在しない
レスポンス	インスタレーション情報 (登録時の応答と同じ)
注意事項	なし

10.3. インスタレーションの更新

説明	インスタレーションを更新する
メソッド	PUT
パス	/1/{tenant_id}/push/installations/{installationId}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● Content-Type: application/json (必須) ● X-Session-Token : セッショントークン (オプション)
リクエストパラメータ	なし
リクエストボディ	インスタレーション情報 (フォーマットはオブジェクトストレージの更新フォーマットと同じ)
レスポンスコード	● 200 OK : 正常更新した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 404 Not Found : 該当するインスタレーションが存在しない ● 415 Unsupported Media Type: Content-Type 不正

レスポンス	更新後のインスタレーション情報（JSON）
注意事項	● インスタレーションを部分更新で想定外のフォーマットに変更してしまった場合、インスタレーションは自動削除され、404 Not Found が返却される。

10.4. インスタレーションの削除

説明	インスタレーションを削除する
メソッド	DELETE
パス	/1/{tenant_id}/push/installations/{installationId}
HTTP ヘッダ	● X-Application-Id: アプリケーション ID（必須） ● X-Application-Key: アプリケーションキー（必須） ● X-Session-Token：セッショントークン（オプション）
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK：正常削除した ● 401 Unauthorized：認証エラー ● 404 Not Found：該当するインスタレーションが存在しない
レスポンス	なし
注意事項	なし

10.5. インスタレーションの検索

説明	インスタレーションを検索する
メソッド	GET
パス	/1/{tenant_id}/push/installations
HTTP ヘッダ	● X-Application-Id: アプリケーション ID（必須） ● X-Application-Key: マスターキー（必須）
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK：正常取得 ● 400 Bad Request：パラメータ/リクエストボディ不正 ● 401 Unauthorized：認証エラー
レスポンス	インスタレーションの配列
注意事項	マスターキーが必要なので、通常のアプリケーションでは使用不可。 SDK では機能を提供しない。

以下のようにインスタレーションの配列が JSON で返却される。

```
{
    "results": [
        {
            "_id": "541bc0a0e4b03de4b0287b2e",
            "_deviceToken": "xxxx",
            ...
        },
        { ... }
    ]
}
```

10.6. Push 送信

説明	Push 通知を送信する
メソッド	POST
パス	/1/{tenant_id}/push/notifications
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: アプリケーションキー (必須) ● Content-Type: application/json (必須)
リクエストパラメータ	なし
リクエストボディ	Push 通知
レスポンスコード	● 200 OK : 正常送信した ● 400 Bad Request : パラメータ/リクエストボディ不正 ● 401 Unauthorized : 認証エラー ● 403 Forbidden : 認可エラー ● 415 Unsupported Media Type: Content-Type 不正
レスポンス	後述
注意事項	● デベロッパコンソールでクライアント Push が禁止されている場合、マスターキーを指定しなければならない。アプリケーションキーを指定した場合は 403 Forbidden エラーとなる。

PUSH メッセージは JSON で以下のように指定する。

```
{
    "query": { /* 送信先指定 */ },
    "message": "Test message",
    "badge": 5,
    "sound": "xxx",
    "content-available": 1,
    "category": "xxx",
    "uri": "xxx",
    "title": "test",
    "sseEventId" : "xxx",
    "sseEventType" : "xxx",
    "allowedReceivers": [ "g:admin" ]
}
```

以下のプロパティは、OS 非依存かつ必須である。

- query : 送信先インスタレーションを指定するためのクエリ (後述)
- message : Push メッセージ本文

以下のプロパティは、OS 非依存かつオプションである。

- allowedReceivers : 通知を受信可能なユーザ・グループの一覧(後述)

以下のプロパティは iOS 固有であり、すべてオプションである。

- badge : バッジカウント
- sound : Application Bundle 内のサウンドファイル名
- content-available : 1 にセットすると、バックグランド Push を有効にする。
- category : Notification カテゴリ

以下のプロパティは Android 固有であり、すべてオプションである。

- title：システムバーに表示するタイトル
- uri：通知を開いたときに起動する URI

以下のプロパティは SSE 固有であり，全てオプションである.
- sseEventId：イベント ID
- sseEventType: イベントタイプ

query には、インスタレーションを指定するための MongoDB クエリ式を指定する。これらのクエリは、インスタレーションオブジェクトに対するクエリとして動作する。以下に例を示す。
例 1) チャネル "chan1" を含むインスタレーションを指定する。

```
{ "_channels": "chan1" }
```

例 2) インスタレーション ID を1つ指定する。

```
{ "_id": "541bc0a0e4b03de4b0287b2e" }
```

例 3) インスタレーション ID を複数指定する。

```
{
    "_id": {
        "$in": ["541bc0a0e4b03de4b0287b2e", "541bc0a0e4b03de4b0287b2f"]
    }
}
```

例 4) E-mail アドレスを指定する（インスタレーション登録時に E-mail が指定されている前提）

```
{ "email": "foo@example.com" }
```

allowedReceivers を使用して、この通知を受信可能なユーザ・グループを制限することができる。このオプションを指定した場合、適合する owner (allowedReceivers に含まれるユーザ ID の1つに一致する、もしくは allowedReceivers に指定されたグループのいずれかに所属する)がセットされているインスタレーションに対してのみ Push が配信される。allowedReceivers に適合しないか、または owner が設定されていないインスタレーションには配信されない。
グループを指定する場合はグループ名の先頭に "g:" を付けて指定する。

例 5) allowedReceivers を指定する。（1つのユーザ ID と グループ "group1" を指定する場合）

```
{ "allowedReceivers": ["55ed50600c69293704e34f9d", "g:group1" ] }
```

allowedReceivers のグループには、"g:anonymous", "g:authenticated" は指定できない。

allowedReceivers を指定しなかった場合は、配信の制限は実施されず、query で指定された全インスタレーションに Push 配信される。

リクエストが成功すると、200 OK が返る。レスポンスには以下のように該当したインスタレーション数が返る。

```
{
    "result": "ok",
    "installations": 17
```

```
}
```

該当するインスタレーションが無い場合は、エラーではなく "installations":0 の結果が返る

エラー時には、"error" にエラーメッセージが返却される。

```
{
    "error": "error messages..."
}
```

11. 管理系API

11.1. サーバ情報取得

説明	モバイルバックエンド基盤のサーバ情報を取得する
メソッド	GET
パス	/1/{tenant_id}/management/server_info
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須) ● X-Application-Key: マスターキー (必須)
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常取得 ● 401 Unauthorized : 認証エラー
レスポンス	バージョン情報を含む JSON オブジェクト
注意事項	なし

以下の例のようにサーバの情報が JSON で返却される。

```
{
    "version": "6.0.0",
    "buildTime": "2016-12-19T09:57:23Z"
}
```

返却されるフィールドは以下のとおり。

- version: バージョン番号
- buildTime: ビルド日時

11.2. ヘルスチェック

説明	モバイルバックエンド基盤のサーバ稼働状態を返す
メソッド	GET
パス	/1/_health
HTTP ヘッダ	なし
リクエストパラメータ	なし
リクエストボディ	なし
レスポンスコード	● 200 OK : 正常取得
レスポンス	稼働状態を含む JSON オブジェクト
注意事項	なし

以下の例のようにサーバの情報が JSON で返却される。JSON の内容は固定である。

```
{
    "name": "api",
    "state": "running"
}
```

11.3. API統計情報取得

説明	API 統計情報を取得する
メソッド	GET
パス	/1/{tenant_id}/management/apistats （マスターキー指定の場合） /1/_system/management/apistats （システムキー指定の場合）
HTTP ヘッダ	● X-Application-Id: アプリケーション ID (必須、ただしシステムキー使用時は不要) ● X-Application-Key: マスターキー または システムキー(必須)
リクエストパラメータ	● total： 積算値フラグ(オプション) ● tenantId： 検索対象をテナント ID で絞り込む(オプション) ● appId： 検索対象をアプリケーション ID で絞り込む(オプション) ● customApiName： 検索対象をカスタム API 名で絞り込む(オプション) ● order： ソート順序 (オプション) ● limit： 検索数上限。デフォルト値は 1000 件。 (オプション) ● start： 開始日時(ISODate 形式) (オプション) ● end： 終了日時(ISODate 形式) (オプション)
リクエストボディ	なし
レスポンスコード	● 200 OK： 正常取得 ● 400 Bad Request： リクエストパラメータ不正 ● 401 Unauthorized： 認証エラー
レスポンス	API 統計情報を含む JSON オブジェクト
注意事項	● システムキー指定の場合のみリクエストパラメータに tenantId を指定可。 ● tenantId と appId の同時指定による検索対象の絞り込みは不可。同時に指定した場合、レスポンスは「400 Bad Request」となる。 ● customApiName を指定する場合は、同時に tenantId もしくは appId の指定が必要。同時に指定されなかった場合、レスポンスは「400 Bad Request」となる。

該当テナントもしくは全テナントを対象とし、MongoDB に 10 分単位で保存している API 統計情報を取得する。結果は以下のように "results" に配列形式で格納される。

例)全テナントを取得対象とした場合

```
{
  "results":[
    {
      "createdAt":"2017-07-12T00:00:000Z",
      "tenantId":"テナント ID_1",
      "アプリケーション ID_1":{
            "users":2,
            "customApi":{
              "hello":{"/sayHello":{"GET":1}}},
            "functions":1
```

```
      },
      "アプリケーション ID_2":{
            "objects":3,
      }
    },
    {
      "createdAt":"2017-07-12T00:00:000Z",
      "tenantId":"テナント ID_2",
    },
    …
}
```

11.3.1. API統計情報の取得範囲

X-Application-Key にマスターキーを指定した場合、該当アプリを含むテナント内の API 統計情報を取得する。システムキーを指定した場合は全テナントの API 統計情報を取得する。

11.3.2. 積算値フラグ (total)

total に true を設定した場合、API 統計情報の積算値を取得する。テナント毎、アプリ毎の積算値も合わせて取得する。total を省略、または、false を指定した場合、10 分単位の API 統計情報を取得する。

API 統計情報の積算値は以下のように、テナント毎"results"に配列形式で格納される。

例)全テナントを取得対象とした場合

```
{
  "results":[
    {
      "tenantId":"テナント ID_1",
      "tenantTotal":15,                                  //テナント毎の積算値
      "アプリケーション ID_1":{
            "appTotal":11,                               //アプリ毎の積算値
            "users":3,                                   ┐
            "customApi":{                                │
              "hello":{"/sayHello":{"GET":2},            ├ API 種別毎の積算値
                       "/tweetHello":{"GET":1}}},        │
            "functions":5                                ┘
      },
      "アプリケーション ID_2":{
        …
      }
    },
    {
      "tenantId":"テナント ID_2",
        …
    },
    …
  ]
}
```

11.3.3. テナントIDでの検索対象の絞り込み (tenantId)

指定されたテナント ID で検索対象の絞り込みを行う。
X-Application-Key にシステムキーを設定している場合のみ指定可能とする。X-Application-Key にマスターキーを設定している場合、レスポンスは「400 Bad Request」となる。

テナント ID_1 で検索対象の絞り込みを行う場合は、以下のように指定する。

```
tenantId=テナント ID_1
```

10 分単位の API 統計情報をテナント ID_1 で絞り込みした場合の例を以下に示す。

```
{
  "results":[
    {
      "createdAt":"2017-07-12T00:00:000Z",
      "tenantId":"テナント ID_1",
      "アプリケーション ID_1":{
             "users":2,
             "customApi":{
               "hello":{"/sayHello":{"GET":1}}},
             "functions":1
      },
      "アプリケーション ID_2":{
             "objects":3,
      }
    },
    {
      "createdAt":"2017-07-12T00:20:000Z",          //API 実行回数 0 のデータも取得対象
      "tenantId":"テナント ID_1"
    }
    …
}
```

11.3.4. アプリケーションIDでの検索対象の絞り込み (appId)

指定されたアプリケーション ID で検索対象の絞り込みを行う。

アプリケーション ID_1 で検索対象の絞り込みを行う場合は、以下のように指定する。

```
appId=アプリケーション ID_1
```

10 分単位の API 統計情報をアプリケーション ID_1 で絞り込みした場合の例を以下に示す。

```
{
  "results":[
    {
      "createdAt":"2017-07-12T00:00:000Z",
      "tenantId":"テナント ID_1",
      "アプリケーション ID_1":{
             "users":2,
              …
```

```
      }
    },
    {
      "createdAt":"2017-07-12T00:10:000Z",
      "tenantId":"テナント ID_1"
      "アプリケーション ID_1":{
             "customApi":{
               "hello":{"/sayHello":{"GET":1},
                        "/tweetHello":{"GET":1}}},
              …
       },
      …
    }
}
```

11.3.5. カスタムAPI名での検索対象の絞り込み (customApiName)

指定されたカスタム API 名で検索対象の絞り込みを行う。カスタム API 名を指定する場合は、同時にテナント ID もしくはアプリケーション ID の指定が必要である。

カスタム API 名「hello」で検索対象の絞り込みを行う場合は、以下のように指定する。

```
customApiName=hello
```

10 分単位の API 統計情報をカスタム API 名「hello」とテナント ID「テナント ID 1」で絞り込みした場合の例を以下に示す。

```
{
  "results":[
    {
      "createdAt":"2017-07-12T00:00:000Z",
      "tenantId":"テナント ID_1",
      "アプリケーション ID_1":{
             "customApi":{
               "hello":{"/sayHello":{"GET":1}}}
      }
    },
    {
      "createdAt":"2017-07-12T00:10:000Z",
      "tenantId":"テナント ID_1"
      "アプリケーション ID_1":{
             "customApi":{
               "hello":{"/sayHello":{"GET":1},
                        "/tweetHello":{"GET":1}}},
       }
      "アプリケーション ID_2":{
             "customApi":{
               "hello":{"/sayHello":{"GET":1}}},
       }
    },
    …
}
```

11.3.6. ソート順序 (order)

order パラメータに createdAt を指定することにより API 統計情報の生成時刻でソートを行う。積算値フラグ total が true の場合は無視される。

以下の例は、生成時刻の古い順でソートする。

```
order=createdAt
```

生成時刻の新しい順にソートする場合は以下のように createdAt の前に "-" を付与する。

```
order=-createdAt
```

order を指定しなかった場合は、デフォルトで生成時刻の古い順でソートされる。

createdAt 以外のキーが order に設定された場合、レスポンスは「400 Bad Request」となる。

11.3.7. 検索数上限 (limit)

返却する検索数の上限を指定する。対象は 10 分単位で保存してある API 統計情報であり、積算値フラグ total が true の場合は無視される。

以下の例では、API 統計情報 500 件を取得する。

```
limit=500
```

limit のデフォルト値は 1000 件とする。limit を指定しなかった場合は、デフォルトでこの値が指定されたものとみなされる。

limit に -1 を指定した場合は制限なし(無限大)として扱う。

注: サーバ側のコンフィグレーションによっては、limit 値に制限がかけられている場合がある。この場合、上限値を越える値を指定したり -1 を指定した場合は 400 Bad Request エラーとなる。

11.3.8. 開始日時 (start)、終了日時(end)

開始日時と終了日時で集計するデータの期間を指定する。開始日時、または、終了日時のみの指定も可。時刻は UTC 時刻で指定する。

例) 2017/04/01 から 2017/05/31 までの統計情報を取得

```
例) 2017/04/01 から 2017/05/31 までの API 統計情報を取得したい場合
start=2017-04-01T00:00:00.000Z&end=2017-05-31T00:00:00.000Z
```