Bintray REST API

General

Access and Versioning

The Bintray REST API is versioned. The latest version of the API will always be available at:

https://api.bintray.com

A specific version can be accessed at:

https://bintray.com/api/v1

Authentication

The Bintray REST API requires an applicative API key. An API key can be obtained from the user profile page. Authentication is achieved using HTTP Basic Authentication with the user's name as username and the API key as the password. Authenticated REST calls should only be used via HTTPs.

Limits

API queries are limited according to the following rules: Limits only apply to resources that do not belong to a user or to the organization the user is part of. The current limit per non-Premium user is 300 queries a day, and 1440 per month. Limits are returned in the `X-RateLimit` headers. For example:

```
X-RateLimit-Limit: 1440
X-RateLimit-Remaining: 257
```

Pagination

Results may be subject to pagination. The current page size is 50 results per page, where only 50 results of a single page are returned by each query. In this case the `start_pos` query parameter may be specified.

In case the pagination is used, the `X-RangeLimit` headers will be included in the response to indicate what results are being viewed.

```
GET /../../..&start_pos=250
```

```
X-RangeLimit-Total: 20000
X-RangeLimit-StartPos: 250
X-RangeLimit-EndPos: 299
```

GPG Signing

Bintray supports automatic content signing for all repositories, and repository metadata signing for specific repository types, such as Debian and YUM.
At the minimum, this requires a GPG public key configured for the repository owner (individual or organization). Additional signing information can also be kept in Bintray or passed via REST.

REST-based signing information may be provided in one of two ways, depending on the GPG information already stored in Bintray:

- Using a `X-GPG-PASSPHRASE` header
  Whenever the private key in Bintray requires a passphrase.

```
X-GPG-PASSPHRASE: :passphrase
```

- Using a JSON body
  Whenever additional signing information is required, such as:
  - Alternative Bintray subject for the GPG public key
  - Optional private key, if not stored in Bintray
  - Optional private key passphrase, if required

```
{
  "subject": "dima",
  "passphrase": "papX***yH8eKw",
  "private_key": "-----BEGIN PGP PRIVATE KEY BLOCK-----"
}
```

When signing information is not provided or is partial no GPG signing will occur.

Content Downloading

Download Content

```
GET https://dl.bintray.com/:subject/:repo/:file_path
GET https://:subject.bintray.com/:repo/:file_path
```

Download content from the specified repository path.

Note: Downloads are done through the `dl.bintray.com` domain, or `:subject.bintray.com` for Premium accounts.

Download only requires using HTTP BASIC Authentication when downloading content from *private repositories* or when downloading *unpublished files*.

Security: Authenticated user with 'read' permission for private repositories, or read entitlement for a repository path.

Dynamic Download

```
GET /content/:subject/:repo/:file_path?bt_package=:package
```

This resource is only available for Bintray Premium repositories.

Download a file based on a dynamic `file_path`.

Currently, only the `$latest` token is supported, which is useful for downloading the latest file published under a specified package.
Package name can be supplied as a:
The `bt_package` query parameter
The `bt_package` matrix parameter
The `X-Bintray-Package` request header

For example:

```
GET /content/bintray/jcenter/com/jfrog/bintray/client/bintray-client-java-api/$lates
```

A successful call will return a 302 redirect to a generated signed URL (with 15 second expiry) to the resolved file path.

```
Status: 302 OK
```

EULA-Protected Product Version download

A user must accept the version's EULA using the access key provided before the first download. An attempt to download the content without accepting the EULA will generate a notification that includes the following response: "To download the resource :artifact you need to accept its EULA at: http://bintray.com/:subject/product/:product/:artifact.version.name /accept_eula?username=:username"
The user should browse to the specified URL, login using the access key provided, and accept the EULA. Subsequent download attempts using the same access key will then succeed.

Security: Authenticated user with 'read' permission for private repositories, or read entitlement for a repository path.

URL Signing

```
POST /signed_url/:subject/:repo/:file_path[?encrypt=:true/false]
```

```
{
  "expiry": 7956915742000,
  "valid_for_secs": 30,
  "callback_id": :id,
  "callback_email": :email,
  "callback_url": :url[?QUERY_PARAM=%callback_id],
  "callback_method": :method,
  "secret": :secret
}
```

This resource is only available to Bintray Premium users.
Generates an anonymous, signed download URL with an expiry date.
Caller must be an owner of the repository or a publisher in the organization owning the repository.
Encrypted download is possible - encryption will be done using AES 256 CBC, see below documentation.

The following input parameters can be specified:
`encrypt`: Optional. Default false. When setting encrypt to true, will use the given secret (below) to encrypt the link response payload. If encrypt is set to true and no secret is given, Bintray will create a secret for you (recommended).
`expiry`: Optional. An expiry date for the URL after which the URL will be invalid, expiry value is in Unix epoch time in milliseconds. By default, expiry date will be 24 hours if not specified. Mutually exclusive with `valid_for_secs`.
`valid_for_secs`: Optional. The number of seconds since generation before the URL expires. Mutually exclusive with `expiry`.
`callback_id`: Optional. An applicative identifier for the request. This identifier appears in download logs and is used in email and download webhook notifications.
`callback_email`: Optional. An email address to send mail to when a user has used the download URL.
This requires a `callback_id`. The `callback_id` will be included in the mail message.
`callback_url`: Optional. A webhook URL to call when a user has used the download URL.
`callback_method`: Optional. HTTP method to use for making the callback; Will use POST if unspecified. Supported methods are: GET, POST, PUT and HEAD.
`secret`: Optional. Secret to be used to encrypt the link response payload.

```
X-Bintray-Secret: The secret to use for decryption - this header will be returned in case enc
Status: 200 OK
{
  "url":"https://dl.bintray.com/:subject/:repo/:file_path?expiry=EXPIRY&id=ENCRYPTED_ID&signa
}
```

Security: Authenticated user with 'publish' permission, or read/write entitlement for a repository path

Callback Identification in Download Logs

The `callback_id` will appear with a colon (`:`) prefix as part of the username field in a download log line.
For example:

Signed url download by an anonymous user with a callback id of 'user254':

```
82.102.172.26 - anonymous:user254 [2014-11-14T23:50:10.207 +0000] "GET /jfrog/artifa
```

Signed url download by an identified bintray user 'kermit' and a callback id of 'user254':

```
82.102.172.26 - kermit:user254 [2014-11-14T23:50:10.207 +0000] "GET /jfrog/artifacto
```

Regular url download by an anonymous user and no callback id:

```
82.102.172.26 - anonymous [2014-11-14T23:50:10.207 +0000] "GET /jfrog/artifactory/ar
```

Callback Identification in Download Webhooks

When a callback is used the webhook payload will include a `callback_id` field.
For example:

```
Status: 200 OK
```

```
"X-Bintray-Callback-Hmac": "Base64 HMAC-MD5 of :subject/:repo/:file_path keyed by the subject
{
  "subject": "my-org",
  "repo": "repo",
  "package": "my-package",
  "version": "1.2.1",
  "file_path": "a/b/release.bin)",
  "callback_id": "malcolm",
  "ip_address": "192.10.2.34"
}
```

Content Uploading & Publishing

Upload Content

```
PUT /content/:subject/:repo/:package/:version/:file_path[?publish=0/1][?override=0/1
```

or:

```
X-Bintray-Package: :package
X-Bintray-Version: :version
[X-Bintray-Publish: 0/1]
[X-Bintray-Override: 0/1]
[X-Bintray-Explode: 0/1]
PUT /content/:subject/:repo/:file_path
```

or:

```
PUT /content/:subject/:repo/:file_path;bt_package=:package;bt_version=:version[;publ
```

Upload content to the specified repository path, with package and version information (both required).
Package and version can be specified in one of the following ways:

1. On the request path
2. As request headers
3. As matrix parameters

Optionally publishing the uploaded artifact(s) as part of uploading (off by default). Additional content can be uploaded to a published version within 180 days from its publish date.

Published artifacts may be redeployed within 180 days; To override an already-published artifact you need to specify "override=1". The override switch can be specified in one of the following ways:

1. As a query parameter
2. As a request header
3. As a matrix parameter

Optionally, supply a `X-Checksum-Sha2` header with client-side sha2 checksum. Bintray will verify the given sha2 and on mismatch will return a 409 Conflict error response.

Security: Authenticated user with 'publish' permission, or read/write entitlement for a repository path

Automatic Signing on Upload

You may supply a passphrase for signing uploaded files and repository metadata using the `X-GPG-PASSPHRASE` header. Please read this for additional details.

Maven Upload

```
PUT /maven/:subject/:repo/:package/:file_path[;publish=0/1]
```

Upload Maven artifacts to the specified repository path, with package information (required). Version information is resolved from the path, which is expected to follow the Maven layout. Optional direct publishing (off by default).

```
Status: 201 Created
{
  "message": "success"
```

```
}
```

Automatic Signing

You may supply a passphrase for signing uploaded files using the `X-GPG-PASSPHRASE` header. Please read this for additional details.

Debian Upload

```
PUT /content/:subject/:repo/:package/:version/:file_path;deb_distribution=:distribut
```

When artifacts are uploaded to a Debian repository using the Automatic index layout, the Debian distribution information is required and must be specified. Information may be specified using HTTP matrix parameters (as shown in the example above). Alternatively, this information can be specified using HTTP request headers:

```
X-Bintray-Debian-Distribution: <DISTRIBUTIONS>
X-Bintray-Debian-Component: <COMPONENTS>
X-Bintray-Debian-Architecture: <ARCHITECTURES>
```

Debian-related parameters accept multiple, comma-separated values. For example:

```
X-Bintray-Debian-Distribution: wheezy
X-Bintray-Debian-Component: main
X-Bintray-Debian-Architecture: i386,amd64
```

In this example, the calculated repository metadata for the uploaded .deb will include the "main" component in "wheezy" distribution in both "i386" and "amd64" architectures.

The ":file_path" for the uploaded .deb may take any value. For example:

```
pool/main/m/mypackage_1.0.0_amd64.deb
```

Automatic Metadata Signing

When uploading with "publish=1" repository metadata will be automatically calculated. Metadata can also be signed automatically. You may supply a passphrase for signing repository metadata using the `X-GPG-PASSPHRASE` header, or trigger repository metadata calculation with additional signing details.

```
Status: 201 Created
{
  "message": "success"
}
```

Publish/Discard Uploaded Content

```
POST /content/:subject/:repo/:package/:version/publish
```

Asynchronously publishes all unpublished content for a user's package version. Returns the number of to-be-published files.
In order to wait for publishing to finish and run this call synchronously, specify a `"publish_wait_for_secs"` timeout in seconds. To wait for the maximum timeout allowed by Bintray use a wait value of `-1`. A wait value of `0` is the default and is the same as running this call asynchronously without waiting.

Optionally, pass in a `"discard"` flag to discard any unpublished content, instead of publishing.

Automatic Signing for Repository Metadata

For repositories that support automatic calculation of repository metadata (such as Debian and YUM), you may supply signing required information, as additional fields on the json body. Please read this for more details.

```
{
  ...optional signing details...
  "discard": true,
  "publish_wait_for_secs": -1
}
```

```
Status: 200 OK
```

```
{
  "files": 39
}
```

In case of a synchronous publish and timeout was reached:

```
Status: 408 Request Timeout
```

Security: When publishing a version, authenticated user with 'publish' permission, or version read/write entitlement. When discarding a version, authenticated user with 'publish' permission, or repository read/write entitlement.

Delete Content

```
DELETE /content/:subject/:repo/:file_path
```

Delete content from the specified repository path, Currently supports only deletion of files.
For OSS, this action is limited for 180 days from the content's publish date.

```
Status: 200 OK
{
  "message": "success"
}
```

Security: Authenticated user with 'publish' permission, or read/write entitlement for a repository path

Content Signing

Get GPG public key

```
GET /users/:user/keys/gpg/public.key
GET /orgs/:org/keys/gpg/public.key
```

Get the subject or organization GPG public key.
The response Content-Type format is 'application/pgp-keys'.

```
Status: 200 OK
"-----BEGIN PGP PUBLIC KEY BLOCK-----\n" +
" ... " +
"-----END PGP PUBLIC KEY BLOCK-----"
```

GPG Sign a Version

```
POST /gpg/:subject/:repo/:package/versions/:version
```

GPG sign all files associated with the specified version.
GPG signing information may be needed. Please refer to this for more details.

```
Status: 200 OK
{
  "message": "success"
}
```

Security: Authenticated user with 'publish' permission, or version read/write entitlement.

GPG Sign a File

```
POST /gpg/:subject/:repo/:file_path
```

GPG sign the specified repository file.
GPG signing information may be needed. Please refer to this for more details.

```
Status: 200 OK
{
  "message": "success"
}
```

Security: Authenticated user with 'publish' permission, or version read/write entitlement.

Content Sync

Sync Version Artifacts to Maven Central

```
POST /maven_central_sync/:subject/:repo/:package/versions/:version
```

```
{
  "username": "userToken", // Sonatype OSS user token
  "password": "passwordToken", // Sonatype OSS user password
  "close": "1" // Optional
}
```

Once Sonatype oss credentials have been set in subject "Accounts" tab, user can send this rest call without specifying username and password (or use different username/password by specifying them in JSON body) Sync version files to a oss.sonatype.org staging repository to publish these files to Maven Central.
By default the staging repository is closed and artifacts are released to Maven Central. You can optionally turn this behaviour off and release the version manually. This is achieved by passing 0 in the 'close' field of the JSON passed to the call.

```
Status: 200 OK
{
  "status": "Successfully synced and closed repo.",
  "messages": ["Sync finished successfully."]
}
```

Security: Authenticated user with 'publish' permission, or a version read/write entitlement.

Repositories

Get Repositories

```
GET /repos/:subject
```

Get a list of repos writable by subject (personal or organizational)

```
Status: 200 OK
[
  {
    "name": "repo",
    "owner": "subject"
  }
]
```

Security: Authenticated user with 'read' permission for private repositories, or repository read entitlement.

Get Repository

```
GET /repos/:subject/:repo
```

Get general information about a repository of the specified user

```
Status: 200 OK
{
  "name": "repo",
  "owner": "user",
  "type": "maven",
  "private": false,
  "premium": false,
  "version_update_max_days": 60,   (only for Enterprise Account, if defined)
  "desc": "This repo...",
  "labels": ["java", "maven"],
  "created": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "package_count": 87
  "gpg_sign_metadata": false,
  "gpg_sign_files":false,
  "gpg_use_owner_key":false
}
```

Security: Authenticated user with 'read' permission for private repositories, or repository read

entitlement.

Create Repository

```
POST /repos/:subject/:repo
```

Create a repository under to the specified subject.
The possible types for a repository are: maven, debian, conan, rpm, docker, npm, opkg, nuget, vagrant and generic (default).

```
{
  "name": "repo",
  "type": "maven",
  "private": false,
  "business_unit": "businessUnit1",
  "desc": "This repo...",
  "labels":["label1", "label2"],
  "gpg_sign_metadata": false,
  "gpg_sign_files":false,
  "gpg_use_owner_key":false,
  "version_update_max_days" : 60    (only for Enterprise Account)
}
```

GPG auto sign flags - the last three flags in the example above are optional, they let you specify whether GPG signing should be applied to this repo. auto signing with gpg is disabled by default.

"business_unit" : a business unit can be associated to repositories allowing you to monitor overall usage per business unit.

"gpg_sign_metadata" : if set to true then the repo's metadata will be automatically signed with Bintray GPG key.

"gpg_sign_files" : if set to true then the repo's files will be automatically signed with Bintray GPG key.

"gpg_use_owner_key" : if set to true then the repo's metadata and files will be signed automatically with the owner's GPG key. this flag cannot be set true simultaneously with either of the bintray key falgs (files or metadata). this flag can be set true only if the repo's owner supplied a private (and public) GPG key on his bintray profile.

"version_update_max_days" : Optional. Number of days after the version is published in which an organization member can upload, override or delete files in the version, delete the version or its package. After this period these actions are not available to the member. This does not apply to the Admin of the repository who can make changes to the version at any time after it is published

Debian repository

it is possible to supply default coordinates upon creation, these coordintes will be used to index the repository thus allowing
the user to include the repository in the sources list while still empty.

```
{
   "type":"debian",
   "default_debian_architecture": "amd64",
   "default_debian_distribution": "jessie"
   "default_debian_component": "main"
}
```

the `default_debian_component` parameter is default "main".

Rpm repository

```
{
  "type": "rpm",
  "yum_metadata_depth": 3,
  "yum_groups_file": "yumGroup.xml" (optional)
}
```

```
Status: 201 Created
{
  "name": "repo",
  "owner": "user",
  "type": "maven",
  "private": false,
  "premium": false,
  "business_unit": "businessUnit1",
  "desc": "This repo...",
  labels:["label1","label2"],
```

```
  "created": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "package_count": 0
}
```

Security: Authenticated user with 'admin' permission.

Update Repository

```
PATCH /repos/:subject/:repo
```

Update a repository under the specified subject

```
{
  "business_unit": "businessUnit1",
  "desc": "This repo..."
  "labels":["label1","label2"]
  "gpg_sign_metadata": false,
  "gpg_sign_files":false,
  "gpg_use_owner_key":false,
  "version_update_max_days" : 60    (only for Enterprise Account)
}
```

```
Status: 200 OK
```

Security: Authenticated user with 'admin' permission.

Delete Repository

```
DELETE /repos/:subject/:repo
```

Delete the specified repository under the specified subject

```
Status: 200 OK
{
  "message": "success"
}
```

Security: Authenticated user with 'admin' permission.

Repository Search

```
GET /search/repos?name=:name&desc=:desc
```

Search for a repository. At least one of the name and desc search fields need to be specified.
Returns an array of results, where elements are similar to the result of getting a single repository.
Search results will not contain private repositories.

Security: Authenticated user is required

Link Package

```
PUT /repository/:subject/:repo/links/:source_subject/:source_repo/:source_package
```

Link the package source_package into the repo repository.
Caller must be an admin of the organization owning the repository.

Optionally accepts a JSON body with path prefix, a specific path to include the files from

```
{
   "path_prefix": "x/y/z"
}
```

```
Status: 201 Created
{
  "message": "success"
}
```

Security: Authenticated user with 'publish' permission, or repository read/write entitlement.

Unlink Package

```
DELETE /repository/:subject/:repo/links/:source_subject/:source_repo/:source_package
```

Unlink the package source_package from the repo repository.
Caller must be an admin of the organization owning the repository.

```
Status: 200 OK
{
  "message": "success"
}
```

Security: Authenticated user with 'publish' permission, or repository read/write entitlement.

Schedule Metadata Calculation

```
POST /calc_metadata/:subject/:repo/[:path]
```

Schedule metadata (index) calculation for the specified repository. For a Maven repository you need to specify the path in the repository for which the metadata should be calculated. For an RPM repository, you need to specify the path according to the repository 'YUM Metadata Folder Depth' field, if different from zero. For other repository types the path is ignored.

Security: Authenticated user with 'publish' permission, or repository read/write entitlement.

Automatic Signing for Repository Metadata

For repositories that support repository metadata signing (such as Debian and YUM), supply signing required information, using one of the methods described here.

```
Status: 202 Accepted
{
  "message": "Calculation was successfully scheduled for '/:subject/:repo'"
}
```

Geo Restrictions

Restrict access to the repository based on the client's Geo location.
This feature is limited to users with Enterprise account.
Enterprise accounts have no limitation on the number of countries specified in their white list or black list.

Note: The country code format is ISO 3166 alpha-2 means each country is defined by 2 letter only.
For example: USA is US, Canada is CA etc.

Get Geo Restrictions

```
GET /repos/:subject/:repo/geo_restrictions
```

Get the list of countries which are defined in the 'black_list' or in the 'white_list'.

```
Status: 200 OK
{
  "white_list" : ["US","CA"],
  "black_list" : []
}
```

Security: Authenticated user with 'admin' permission.

Update Geo Restrictions

```
PUT /repos/:subject/:repo/geo_restrictions
```

Update the 'black_list' or 'white_list' with the related countries code.

Note: The update can be done on one list only.

```
{
  "white_list" : ["US", "CA"],
  "black_list" : []
}
```

```
Status: 200 OK
{
  "The Geo Restrictions for repo path :subject/:repo were updated successfully"
}
```

Security: Authenticated user with 'admin' permission.

Delete Geo Restrictions

```
DELETE /repos/:subject/:repo/geo_restrictions
```

Remove all the countries from the 'white_list' and 'black_list'.

```
Status: 200 OK
{
  "white_list" : [],
  "black_list" : []
}
```

Security: Authenticated user with 'admin' permission.

IP Restrictions

Restrict download of files from a repository based on source ip. Limitation is based on CIDR notation.

Get IP Restrictions

```
GET repos/:subject/:repo/ip_restrictions
```

Gets whitelisted and blacklisted CIDRs.

```
Status: 200 OK
{
  "white_cidrs": [
    "10.0.0.1/32"
  ],
  "black_cidrs": []
}
```

Set IP Restrictions

```
PUT repos/:subject/:repo/ip_restrictions
```

Update ip restrictions with the given white list and black list of CIDRs.

```
{
  "white_cidrs": [
    [
      "10.0.0.1/32",
      "10.0.0.7/32"
    ]
  ],
  "black_cidrs": []
}
```

```
Status: 200 OK
{
  "white_cidrs": [
    "10.0.0.1/32"
  ],
  "black_cidrs": []
}
```

Update IP Restrictions

```
PATCH repos/:subject/:repo/ip_restrictions
```

Add or remove CIDRs from black/white list restrictions.

```
{
  "add": {
    "white_cidrs": [
      "10.0.0.1/32",
      "10.0.0.7/32"
    ],
    "black_cidrs": []
  },
  "remove": {
    "white_cidrs": [
      "10.0.0.9/32",
```

```
      "10.0.0.6/24"
    ],
    "black_cidrs": [
      "10.100.0.9/16"
    ]
  }
}
```

```
Status: 200 OK
{
  "white_cidrs": [
    "10.0.0.7/32",
    "10.0.0.1/32"
  ],
  "black_cidrs": []
}
```

Delete IP Restrictions

```
DELETE repos/:subject/:repo/ip_restrictions
```

Removes all restrictions, black and white.

```
Status: 200 OK
{
  "message": "Successfully deleted restriction for repo /:subject/:repo"
}
```

Packages

Get Packages

```
GET /repos/:subject/:repo/packages[?start_pos=122][&start_name=prefix]
```

Get a list of packages in the specified repository, optionally specify a starting position and/or a name prefix filter This resource can be consumed by both authenticated and anonymous clients. For anonymous clients it will return no more than 50 results

```
Status: 200 OK
[
  {
    "name": "package1",
    "linked": false
  }
]
```

Security: Authenticated user with 'read' permission, or repository read entitlement.

Get Package

```
GET /packages/:subject/:repo/:package[?attribute_values=1]
```

Get general information about a specified package with package name.

```
Status: 200 OK
{
  "name": "my-package",
  "repo": "repo",
  "owner": "user",
  "desc": "This package...",
  "labels": ["persistence", "database"],
  "attribute_names": ["licenses", "vcs", "github", ...], (hidden when using 'attribute_values
  "licenses": ["Apache-2.0"],
  "custom_licenses": ["my-license-1", "my-license-2"],   (only for Premium Account)
  "followers_count": 82,
  "created": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "website_url": "http://jfrog.com",
  "rating": 8,
  "issue_tracker_url": "https://github.com/bintray/bintray-client-java/issues",
  "linked_to_repos": [],
  "github_repo": "", (publishers only)
  "github_release_notes_file": "", (publishers only)
  "public_download_numbers": false, (publishers only)
  "public_stats": true, (publishers only)
  "permissions": [],
```

```
  "versions": ["0.9", "1.0", "1.0.1", ...],
  "latest_version": "1.2.5",
  "rating_count": 8,
  "system_ids" : [],
  "updated": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "vcs_url": "https://github.com/bintray/bintray-client-java.git",
  "attributes": "{"attr1_name":["attr1_value"], "attr2_name":["attr2_value"]}"   (only when '
}
```

Security: Non-authenticated user.

Create Package

```
POST /packages/:subject/:repo
```

Creates a new package in the specified repo (user has to be an owner of the repo)
`licenses` and `vcs_url` are mandatory for OSS packages.
The value of `licenses` needs to be one of a predefined set of licenses
`custom_licenses` is available only for Premium accounts and supports referencing custom, proprietary licenses associated with the subject owning the package.
`public_stats` is available only for Premium accounts.

```
{
  "name": "my-package",
  "desc": "This package...",
  "labels": ["persistence", "database"],
  "licenses": ["Apache-2.0", "GPL-3.0"],
  "custom_licenses": ["my-license-1", "my-license-2"],
  "vcs_url": "https://github.com/bintray/bintray-client-java.git",
  "website_url": "http://jfrog.com",
  "issue_tracker_url": "https://github.com/bintray/bintray-client-java/issues",
  "github_repo": "bintray/bintray-client-java",
  "github_release_notes_file": "RELEASE.txt",
  "public_download_numbers": false,
  "public_stats": true
}
```

```
Status: 201 OK
{Package get JSON response}
```

Security: Authenticated user with 'publish' permission, or repository read/write entitlement.

Delete Package

```
DELETE /packages/:subject/:repo/:package
```

Delete the specified package

```
Status: 200 OK
{
  "message": "success"
}
```

Security: Authenticated user with 'publish' permission, or repository read/write entitlement.

Update Package

```
PATCH /packages/:subject/:repo/:package
```

Update the information of the specified package.
The value of `licenses` needs to be one of a predefined set of licenses
`custom_licenses` is available only for Premium accounts and supports referencing custom, proprietary licenses associated with the subject owning the package.
`public_stats` is available only for Premium accounts.

```
{
  "desc": "This package...",
  "labels": ["persistence", "database"],
  "licenses": ["Apache-2.0", "GPL-3.0"],
  "custom_licenses": ["my-license-1", "my-license-2"],
  "vcs_url": "https://github.com/bintray/bintray-client-java.git",
  "website_url": "http://jfrog.com",
  "issue_tracker_url": "https://github.com/bintray/bintray-client-java/issues",
  "github_repo": "bintray/bintray-client-java",
  "github_release_notes_file": "RELEASE_1.2.3.txt",
  "public_download_numbers": false,
```

```
  "public_stats": true
}
```

```
Status: 200 OK
```

Security: Authenticated user with 'publish' permission, or repository read/write entitlement.

Package Search

```
GET /search/packages[?name=:name&desc=:desc&subject=:subject&repo=:repo]
```

Search for a package. At least one of the search fields needs to be specified. subject name and repo name need to be exact. Returns an array of results, where elements are similar to the result of getting a single package.
Search results will not contain private packages.

Security: Non-authenticated user.

Maven Package Search

```
GET /search/packages/maven?g=:groupId&a=:artifactId&q=:query[&subject=:subject&repo=
```

Search for a Maven package using Maven groupId and artifactId. At least one of the Maven coordinates or a wildcard query needs to be specified. subject name and repo name need to be exact.
Returns an array of results in the following format:

```
{
    "name": "test-package",
    "repo": "jcenter",
    "owner": "bintray",
    "desc": "This package....",
    "system_ids": [
      "groupid:artifactid"
    ],
    "versions": [
        1.0,
        2.0
    ],
    "latest_version": "2.0"
}
```

The system_ids field contains matched groupId:artifactId coordinates under each found package.
Search results will not contain private packages.

Example usage:

```
GET https://api.bintray.com/search/packages/maven?g=com.jfrog.bintray.gradle&a=*bintray*&subj
```

or:

```
GET https://api.bintray.com/search/packages/maven?q=*bintray*&subject=jfrog&repo=jfrog-jars
```

Security: Non-authenticated user.

Package for File

```
GET /file_package/:subject/:repo/:filePath
```

Get general information about the package a repository file is associated with.
Returns a response similar to the result of getting a single package.

Security: Non-authenticated user.

Versions

Get Version

```
GET /packages/:subject/:repo/:package/versions/:version[?attribute_values=1]
GET /packages/:subject/:repo/:package/versions/_latest[?attribute_values=1]
```

Get general information about a specified version, or query for the latest version that has at least one file published to it.

```
Status: 200 OK
{
  "name": "1.1.5",
  "desc": "This version...",
  "package": "my-package",
  "repo": "repo",
  "owner": "user",
  "labels": ["OSS", "org-name", ...],
  "published": "true",
  "attribute_names": ["licenses", "vcs", "github", ...],
  "created": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "updated": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "released": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "github_release_notes_file": "", (publishers only)
  "github_use_tag_release_notes": "", (publishers only)
  "vcs_tag": "", (publishers only)
  "ordinal": 5,
  "attributes": "{"attr1_name":["attr1_value"], "attr2_name":["attr2_value"]}" (only when 'at
}
```

Security: Authenticated user with 'read' permission for private repositories, or version read entitlement.

Create Version

```
POST /packages/:subject/:repo/:package/versions
```

Creates a new version in the specified package (user has to be owner of the package)

```
{
  "name": "1.1.5",
  "released": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)", (optional)
  "desc": "This version...",
  "github_release_notes_file": "RELEASE.txt", (optional)
  "github_use_tag_release_notes": true, (optional)
  "vcs_tag": "1.1.5" (optional)
}
```

```
Status: 201 Created
{Version get JSON response}
```

Security: Authenticated user with 'publish' permission, or package read/write entitlement.

Delete Version

```
DELETE /packages/:subject/:repo/:package/versions/:version
```

Delete the specified version

```
Status: 200 OK
{
  "message": "success"
}
```

Security: Authenticated user with 'publish' permission, or package read/write entitlement.

Update version

```
PATCH /packages/:subject/:repo/:package/versions/:version
```

Update the information of the specified version

```
{
  "desc": "This package...",
  "github_release_notes_file": "RELEASE_1.2.3.txt",
  "github_use_tag_release_notes": true,
  "vcs_tag": "1.1.5",
  "released": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)"
}
```

```
Status: 200 OK
```

Security: Authenticated user with 'publish' permission, or package read/write entitlement.

Version for File

```
GET /file_version/:subject/:repo/:filePath
```

Get general information about the version a repository file is associated with.
Returns a response similar to the result of getting a single version.

Security: Non-authenticated user.

Product

This resource is only available to Bintray Enterprise Edition users.

Get Products

```
GET /products/:subject
```

Get a list of products for the specified subject.

```
Status: 200 OK
{
  "name": "productName"
}
```

Security: Authenticated user with 'read' permission for private repositories, or repository read entitlement.

Get Product

```
GET /products/:subject/:product
```

Get details for the specified product.

```
Status: 200 OK
{
 "name": "productName",
 "display_name": "productName",
 "created": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
 "desc": "description",
 "website_url": "http://great-prod.io",
 "vcs_url": "",
 "packages": ["registry/my-docker", "rpms/my-rpm","deb/my-deb"...]
 "versions": ["1.0","1.1"...]
}
```

Security: Authenticated user with 'read' permission for private repositories, or repository read entitlement.

Create Product

```
POST /products/:subject
```

Create a product for the given subject.

```
{
 "name": "productName",
 "display_name": "productName",
 "desc": "description",
 "website_url": "http://great-prod.io",
 "vcs_url": "",
 "packages": ["registry/my-docker", "rpms/my-rpm","deb/my-deb"...],
 "sign_url_expiry": 10
}
```

Status: 201 Created
Body: JSON of the created product

Security: Authenticated user with 'admin' permission.

Update Product

```
PATCH /products/:subject/:product
```

Update an existing product. At least one of the fields must be specified.

```
{
 "display_name": "productName",
 "desc": "description",
 "website_url": "http://great-prod.io",
 "vcs_url": "",
 "packages": ["registry/my-docker", "rpms/my-rpm","deb/my-deb"...]
}
```

```
Status: 200 OK
```

Security: Authenticated user with 'admin' permission.

Delete Product

```
DELETE /products/:subject/:product
```

Delete the specified product and all its sub-elements (such as EULAs).

```
Status: 200 OK
{"message": "deleted"}
```

Security: Authenticated user with 'admin' permission.

EULA Tracking

Get Product Signed EULAs

```
GET /products/:subject/:product/signed_eulas[?from=:fromDate&to=:toDate&username=:us
```

Get a list of users who signed eula per product with sign date, version signed and eula name.

The following query parameters can be specified:
`pagination`: Optional.
`from` and `to`: Optional. dates to filter by, can be supplied separately or together, defined in the following ISO8601 format (yyyy-MM-dd'T'HH:mm:ss.SSSZ).
`username`: Optional. filter by username.
`eula_name`: Optional. filter by Eula name.

```
Status: 200 OK
{
  "artifactory-pro": [
    {
      "date_signed": "2016-11-03T16:00:32.854Z",
      "username": "joebloggs",
      "product_version_name": "1",
      "eula_name": "artifact_pro_eula"
    },
    .
    .
    .
    {
      "date_signed": "2016-11-01T16:00:32.854Z",
      "username": "jainbloggs",
      "product_version_name": "1",
      "eula_name": "artifact_pro_eula"
    }
  ]
}
```

Security: Authenticated user with 'admin' permission.

Get Signed EULAs - All Products

```
GET /products/:subject/_all/signed_eulas[?from=:fromDate&to=:toDate&username=:userna
```

Get a list of users who signed eula with sign date, version signed and eula name for each product owned by the given subject.

Query parameters are explained here

```
Status: 200 OK
```

```
{
  "product_1": [
                    {
                      "date_signed": "2016-11-06T09:22:34.161Z",
                      "username": "btuser1",
                      "product_version_name": "versions-0",
                      "eula_name": "eula1"
                    },
                    {
                      "date_signed": "2016-11-06T09:22:32.585Z",
                      "username": "btuser2",
                      "product_version_name": "versions-1",
                      "eula_name": "eula2"
                    },
                    {
                      "date_signed": "2016-11-06T09:22:30.492Z",
                      "username": "btuser3",
                      "product_version_name": "versions-2",
                      "eula_name": "eula1"
                    }
                ],
                .
                .
                .
  "product_n": [
                    {
                      "date_signed": "2016-11-06T09:31:33.704Z",
                      "username": "btuser5",
                      "product_version_name": "1.0",
                      "eula_name": "eula8"
                    }
                    {
                      "date_signed": "2016-11-06T09:22:32.585Z",
                      "username": "btuser4",
                      "product_version_name": "versions-1",
                      "eula_name": "eula1"
                    },
                    {
                      "date_signed": "2016-11-06T09:22:30.492Z",
                      "username": "btuser7",
                      "product_version_name": "versions-6",
                      "eula_name": "eula6"
                    }
                ]
}
```

Security: Authenticated user with 'admin' permission.

EULAs

This resource is only available to Bintray Enterprise users.

Get EULAs

```
GET /products/:subject/:product/eulas
```

Get a list of EULAs for the specified product.

```
Status: 200 OK
[
  {
   "name": "EULA name",
   "created": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
   "versions": ["1.0","1.1"...]
  }
]
```

Security: Authenticated user with 'read' permission for private repositories, or repository read entitlement.

Get EULA

```
GET /products/:subject/:product/eulas/:eula
```

Returns the specified product EULA.

```
Status: 200 OK
{
 "name": "EULA name",
 "created": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
```

```
 "versions": ["1.0","1.1"...]
 "syntax": "markdown", [markdown/asciidoc/plain_text default markdown]
 "content": "..."
}
```

Security: Authenticated user with 'admin' permission.

Create EULA

```
POST /products/:subject/:product/eulas
```

Create a EULA for the given subject, with the given product. A new EULA will apply to all new versions if the 'default' parameter is specified as true.

```
{
 "name": "EULA name",
 "syntax": "markdown", [markdown/asciidoc/plain_text default markdown]
 "content": "...",
 "default": false,
 "versions": ["1.0","1.1"...]
}
```

Status: 201 Created
Body: JSON of the created eula

Security: Authenticated user with 'admin' permission.

Update EULA

```
PATCH /products/:subject/:product/eulas/:eula
```

Update a EULA under a specified subject and product. At least one of the fields must be specified.

```
{
 "syntax": "markdown", [markdown/asciidoc/plain_text default markdown]
 "content": "...",
 "default": true,
 "versions": ["1.0","1.1"...]
}
```

```
Status: 200 OK
```

Security: Authenticated user with 'admin' permission.

Delete EULA

```
DELETE /products/:subject/:product/eulas/:eula
```

Delete the specified EULA under the specified subject and product.

```
Status: 200 OK
{"message": "deleted"}
```

Security: Authenticated user with 'admin' permission.

Get EULA by File Path

```
GET /file_eula/:subject/:repo/:file_path
```

Returns the EULA for a specified product by artifact path.

```
Status: 200 OK
Body: JSON of the requested <<url_eula_get,eula>>
```

Security: Authenticated user with 'admin' permission.

Files

Get Package Files

```
GET /packages/:subject/:repo/:package/files[?include_unpublished=0/1]
```

Get all files in a given package.
When called by a user with publishing rights on the package, includes unpublished files in the list. By default only published files are shown.

```
Status: 200 OK
[
  {
    "name": "nutcracker-1.1-sources.jar",
    "path": "org/jfrog/powerutils/nutcracker/1.1/nutcracker-1.1-sources.jar",
    "package": "jfrog-power-utils",
    "version": "1.1",
    "repo": "jfrog-jars",
    "owner": "jfrog",
    "created": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
    "size": 1234,
    "sha1": "602e20176706d3cc7535f01ffdbe91b270ae5012"
  }
]
```

Security: Authenticated user with 'read' permission for private repositories, or package read entitlement.

Get Version Files

```
GET /packages/:subject/:repo/:package/versions/:version/files[?include_unpublished=0
```

Get all files in a given version.
Returns an array of results, where elements are similar to the result of getting package files.
When called by a user with publishing rights on the package, includes unpublished files in the list. By default only published files are shown.

Security: Authenticated user with 'read' permission for private repositories, or version read entitlement.

File Search by Name

```
GET /search/file?name=:name[&subject=:subject&repo=:repo&start_pos=:start_pos
                            &created_after=:dateCreatedAfter]
```

Search for a file by its name. name can take the * and ? wildcard characters. May take an optional subject and/or repo name to search in and/or created_after search from the 'dateCreatedAfter' until today. The 'dateCreatedAfter' is defined in the following ISO8601 format (yyyy-MM-dd'T'HH:mm:ss.SSSZ). Returns an array of results, where elements are similar to the result of getting package files.
Search results will not contain private files.

Security: Authentication is not required

File Search by Checksum

```
GET /search/file?sha1=:sha1[&subject=:subject&repo=:repo&start_pos=:start_pos]
```

Search for a file by its sha1 checksum. May take an optional subject and/or repo name to search in. Returns an array of results, where elements are similar to the result of getting package files.
Search results will not contain private files.

Security: Authentication is not required

File in Download List

```
PUT /file_metadata/:subject/:repo/:file_path
```

Add or remove a file from/to the 'Download List'. Pass true to add the file to the download list, and false to remove it.

```
{
  "list_in_downloads":true
}
```

```
Status: 200 OK
{"message": "success"}
```

Security: Authenticated user with 'publish' permission, or version read/write entitlement.

Readme

Get Readme

```
GET /packages/:subject/:repo/:package/readme
```

Returns the readme for the specified package by subject. Either Bintray readme or GitHub readme.

```
Status: 200 OK
{
  "package": "my-package",
  "repo": "repo",
  "owner": "user",
  "bintray": {
    "syntax": "markdown", [markdown/asciidoc/plain_text default markdown]
    "content": "the quick brown fox"
  },
  OR:
  "github": {
    "github_repo": "gh_user/repo",
  }
}
```

Security: Authenticated user with 'read' permission for private repositories, or package read entitlement.

Create Readme

```
POST /packages/:subject/:repo/:package/readme
```

Creates a new readme for the specified package by subject. "content" has to be passed to the command if using "bintray", or will be retrieved from a GitHub repository, when using "github". GitHub repository name has to be provided.

```
{
  "bintray": {
    "syntax": "markdown", [markdown/asciidoc/plain_text default markdown]
    "content": "the quick brown fox"
  },
OR:
  "github": {
    "github_repo": "gh_user/repo",
  }
}
```

```
Status: 200 CREATED
{
  "package": "my-package",
  "repo": "repo",
  "owner": "user",
  "bintray": {
    "syntax": "markdown", [markdown/asciidoc/plain_text default markdown]
    "content": "the quick brown fox"
  },
  OR:
  "github": {
    "github_repo": "gh_user/repo",
  }
}
```

Security: Authenticated user with 'publish' permission, or package read/write entitlement.

Create Product Readme

```
POST /products/:subject/:product/readme
```

Sets the readme for all of a product's underlying packages. Has a similar effect to setting a package readme, and uses the same syntax.

Security: Authenticated user with 'publish' permission.

Delete Product Readme

```
DELETE /products/:subject/:product/readme
```

Deletes the readme for all of a product's underlying packages.
Status: 200 OK

Security: Authenticated user with 'publish' permission.

Release Notes

Release notes manipulations can be applied to a package or to a version.

Get Package's Release Notes

```
GET /packages/:subject/:repo/:package/release_notes
```

Returns the release notes for a specific package by subject; Either Bintray release notes or GitHub release notes.

```
Status: 200 OK
{
  "package": "my-package",
  "repo": "repo",
  "owner": "user",
  "bintray": {
    "syntax": "markdown", [markdown/asciidoc/plain_text default markdown]
    "content": "the quick brown fox"
  },
  OR:
  "github": {
    "github_repo": "gh_user/repo",
    "github_release_notes_file": "1.2/release_notes.md"
  }
}
```

Security: Authenticated user with 'read' permission for private repositories, or package read entitlement.

Create Package's Release Notes

```
POST /packages/:subject/:repo/:package/release_notes
```

Create release notes for a package by subject; Release notes "content" has to be passed to the command if using "bintray", or will be copied from the provided GitHub release notes if using "github". GitHub repository name has to be provided.
Release notes will be applied to all versions.

```
{
  "bintray": {
    "syntax": "markdown", [markdown/asciidoc/plain_text default markdown]
    "content": "the quick brown fox"
  },
  OR:
  "github": {
    "github_repo": "gh_user/repo",
    "github_release_notes_file": "1.2/release_notes.md"
  }
}
```

```
Status: 200 CREATED
{
  "package": "my-package",
  "repo": "repo",
  "owner": "user",
  "bintray": {
    "syntax": "markdown", [markdown/asciidoc/plain_text default markdown]
    "content": "the quick brown fox"
  },
  OR:
  "github": {
    "github_repo": "gh_user/repo",
   "github_release_notes_file": "1.2/release_notes.md"
  }
}
```

Security: Authenticated user with 'publish' permission, or package read/write entitlement.

Delete Package's Release Notes

```
DELETE /packages/:subject/:repo/:package/release_notes
```

Deletes release notes for a specific package by subject.

```
Status: 200 OK
{"message":"success"}
```

Security: Authenticated user with 'publish' permission, or package read/write entitlement.

Get Version's Release Notes

```
GET /packages/:subject/:repo/:package/versions/:version/release_notes
```

Returns release notes for a specific version by subject; Either Bintray release notes or GitHub release notes.
Note: If using GitHub, the GitHub repository name has to be provided prior to calling this command. This can be done via UI, or by calling update package API.

```
Status: 200 OK
{
  "version": "1.2"
  "package": "my-package",
  "repo": "repo",
  "owner": "user",
  "bintray": {
    "syntax": "markdown", [markdown/asciidoc/plain_text default markdown]
    "content": "the quick brown fox"
  },
  OR:
  "github": {
    "github_release_notes_file": "1.2/release_notes.md"
    OR:
    "use_release": "true"
  }
 }
```

Security: Authenticated user with 'read' permission for private repositories, or version read entitlement.

Create Version's Release Notes

```
POST /packages/:subject/:repo/:package/versions/:version/release_notes
```

Create release notes for a specific version by subject; Release notes "content" has to be passed to the command if using "bintray", or will be copied from the provided GitHub release notes if using "github".
If passing "github", pass either release notes file path into "github_release_notes_file", or "true" to "use_release" to use the release tag as release notes.
Note:
If using GitHub, the GitHub repository name has to be provided prior to calling this command. This can be done via UI, or by calling update package API.
If using "use_release", the vcs_tag should be configured prior to the call of this command. This can be done via UI, or by calling update version API.

```
{
  "bintray": {
    "syntax": "markdown", [markdown/asciidoc/plain_text default markdown]
    "content": "the quick brown fox"
  },
OR:
  "github": {
    "github_release_notes_file": "1.2/release_notes.md"
    OR:
    "use_release": "true"
  }
}
```

```
Status: 200 CREATED
{
  "version": "1.2"
  "package": "my-package",
  "repo": "repo",
  "owner": "user",
  "bintray": {
```

```
    "syntax": "markdown", [markdown/asciidoc/plain_text default markdown]
    "content": "the quick brown fox"
  },
  OR:
  "github": {
    "github_release_notes_file": "1.2/release_notes.md"
    OR:
    "use_release": "true"
  }
}
```

Security: Authenticated user with 'publish' permission, or version read/write entitlement.

Attributes

Get Attributes

```
GET /packages/:subject/:repo/:package/attributes[?names=:att1,:att2]
GET /packages/:subject/:repo/:package/versions/:version/attributes[?names=:att1,:att
```

Get attributes associated with the specified package or version. If no attribute names are specified, return all attributes.

```
Status: 200 OK
[
  {"name": "att1", "values" : ["val1"], "type": "string"},
  {"name": "att2", "values" : [1, 2.2, 4], "type": "number"},
  {"name": "att3", "values" : ["2011-07-14T19:43:37+0100", "2011-07-14T19:43:37+0100", "1994-
]
```

Note: Dates are defined in ISO8601 format.

Security: Authenticated user with 'read' permission for private repositories, or version/package read entitlement for the corresponding calls.

Set Attributes

```
POST /packages/:subject/:repo/:package/attributes
POST /packages/:subject/:repo/:package/versions/:version/attributes
```

Associate attributes with the specified package or version, overriding all previous attributes. Optionally, specify an attribute type. Otherwise, type will be inferred from the attribute's value. If a type cannot be inferred, string type will be used. Non-homogeneous arrays are not accepted. Attributes names beginning with an underscore ("_") will only be visible for users with publish rights. Attribute types can be one of the following: string, date, number, boolean, version version currently behaves like string. This will change with future Bintray versions.

```
[
  {"name": "att1", "values" : ["val1"], "type": "string"}, //string
  {"name": "att2", "values" : [1, 2.2, 4]}, //number
  {"name": "att3", "values" : ["2011-07-14T19:43:37+0100", "2011-07-14T19:43:37+0100", "1994-
  {"name": "att4", "values" : [1.1, "elephant", 3.1]}, //BAD REQUEST
]
```

```
Status: 200 OK
```

Security: Authenticated user with 'publish' permission for private repositories, or version/package read/write entitlement for the corresponding calls.

Update Attributes

```
PATCH /packages/:subject/:repo/:package/attributes
PATCH /packages/:subject/:repo/:package/versions/:version/attributes
```

Update attributes associated with the specified package or version. Attributes may have a null value. Optionally, specify an attribute type. Otherwise, type will be inferred from the attribute's value. If a type cannot be inferred, string type will be used. Non-homogeneous arrays are not accepted. Attribute types can be one of the following: string, date, number, boolean, version version currently behaves like string. This will change with future Bintray versions.

```
[
  {"name": "att1", "values" : ["val1"], "type": "string"}, //string
```

```
  {"name": "att2", "values" : [1, 2.2, 4]}, //number
  {"name": "att3", "values" : ["2011-07-14T19:43:37+0100", "2011-07-14T19:43:37+0100", "1994-
  {"name": "att4", "values" : [1.1, "elephant", 3.1]} //string
]
```

```
Status: 200 OK
```

Security: Authenticated user with 'publish' permission for private repositories, or version/package read/write entitlement for the corresponding calls.

Delete Attributes

```
DELETE /packages/:subject/:repo/:package/attributes[?names=:att1,:att2]
DELETE /packages/:subject/:repo/:package/versions/:version/attributes[?names=:att1,:
```

Delete attributes associated with the specified repo, package or version. If no attribute names are specified, delete all attributes

```
Status: 200 OK
{
  "message": "success"
}
```

Security: Authenticated user with 'publish' permission for private repositories, or version/package read/write entitlement for the corresponding calls.

Attribute Search

```
POST /search/attributes/:subject/:repo[?attribute_values=1]
```

```
POST /search/attributes/:subject/:repo/:package/versions[?attribute_values=1]
```

Search for packages/versions inside a given repository matching a set of attributes. The AND operator will be used when using multiple query clauses, for example attribute A equals X and attribute B is greater than Z When an array value is used, if the existing attribute value is a scalar match against one of the array values; if the existing attribute value is an array check that the existing array contains the query array.

Returns an array of results, where elements are similar to the result of getting a single package or a single version, accordingly.

Note: The values range is defined by the brackets direction and the comma position.

```
[
  {"att1" : ["val1", "val2"]},             // att1 == val1 || att1 == val2           (Relevai
  {"att2": "[1,3]"},                       // 1 <= att2 <= 3                         (Relevai
  {"att3": "[,3]"},                        // att3 <= 3                              (Relevai
  {"att4": "[,3["},                        // att4 < 3                               (Relevai
  {"att5": "]2011-07-14T19:43:37+0100,]"}  // att5 value is after 2011-07-14T20:43:37 (Relevai
  (Dates are defined in ISO8601 format.)
]
```

```
Status: 200 OK
[
  {
    package or version object
  },
  {
    package or version object
  }
]
```

Security: Authenticated user with 'publish' permission for private repositories, or version/package read/write entitlement for the corresponding calls.

File Attributes

File Attributes are associated with the specified Subject Repo and Artifact.
Each Attribute contains 3 fields: 'name', 'type', 'values'. The 'name' and the 'values' are mandatory fields.
Attribute types can be one of the following: STRING, DATE, NUMBER. If the JSON does not include Attribute type, the type will be inferred from the attribute's value. If the type cannot be inferred, string type will be used.
Non-homogeneous arrays are not accepted, means all values should be of the same Type (STRING,

NUMBER, DATE).
Attribute names beginning with an underscore ("_") will only be visible for users with publish rights.

Note: Dates are defined in ISO8601 format.

Get File Attributes

```
GET  /files/:subject/:repo/:file_path/attributes
```

Returns all the attributes related to Artifact. This resource can be consumed by both authenticated and anonymous users.

```
Status: 200 OK
[
  {"name": "att1", "type": "string", "values" : ["val1"]},
  {"name": "att2", "type": "number",  "values" : [1, 2.2, 4] },
  {"name": "att3", "type": "date", "values" : ["2011-07-14T19:43:37+0100",  "2011-07-14T19:43
]
```

Security: Authenticated user with 'read' permission, or repository read entitlement for repository path.

Set File Attributes

```
POST  /files/:subject/:repo/:file_path/attributes
```

Set attributes associated with the specified Artifact. Overriding all previous attributes.

```
Status: 200 OK
[
  {"name": "att1", "type": "string", "values" : ["val1"] }, //string
  {"name": "att2", "values" : [1, 2.2, 4]}, //number
  {"name": "att3", "type": "date", "values" : ["2011-07-14T19:43:37+0100", "2011-07-14T19:43:
]
```

Security: Authenticated user with 'publish' permission, or write entitlement for repository path.

Update File Attributes

```
PATCH  /files/:subject/:repo/:file_path/attributes
```

Update the Artifact with new attributes without removing the older Artifact's attributes.

```
Status: 200 OK
[
  {"name": "att1", "type": "string", "values" : ["val1"]}, //string
  {"name": "att2", "values" : [1, 2.2, 4]}, //number
  {"name": "att3", “type”:”date”, "values" : ["2011-07-14T19:43:37+0100", "2011-07-14T19:43:3
]
```

Security: Authenticated user with 'publish' permission, or write entitlement for repository path.

Delete File Attributes

```
DELETE /files/:subject/:repo/:file_path/attributes[?names=:attr_name_1,:attr_name_2]
```

Remove attributes associated with the specified Artifact. By default, delete all attributes related to the specified Artifact. The 'names' parameter is optional, and is used to remove specific attributes only.

```
Status: 200 OK
{
  "message": "success"
}
```

Security: Authenticated user with 'publish' permission, or write entitlement for repository path.

Search File Attributes

```
POST /files/:subject/:repo/search/attributes
```

Returns all artifacts in the specified repository that at least one of their attributes correspond to names and values specified in the JSON payload.

Note: The values range is defined by the brackets direction and the comma position.

```
[
  {"att1" : ["val1", "val2"]},            // att1 == val1 || att1 == val2           (Releva
  {"att2": "[1,3]"},                      // 1 <= att2 <= 3                         (Releva
  {"att3": "[,3]"},                       // att3 <= 3                              (Releva
  {"att4": "[,3["},                       // att4 < 3                               (Releva
  {"att5": "]2011-07-14T19:43:37+0100,]"} // att5 value is after 2011-07-14T20:43:37 (Releva
  (Dates are defined in ISO8601 format.)
]
```

```
Status: 200 OK
[
  {
    "Name": :file_name,
    “Path”: :file_pat,
    "repo": :repo_name,
    "Package": :package_name,
    "Version": :version,
    "owner": :owner_username,
    "created": :date_created, "size":1,
    "Sha1": :sha1_hash_code,
    "Sha256": :sha256_hash_code
  }
]
```

Security: Authenticated user with 'read' permission, or repository read entitlement for repository path.

Users & Organizations

Get User

```
GET /users/:user
```

Get information about a specified user

```
Status: 200 OK
{
  "name": "user",
  "full_name": "First M. Last",
  "gravatar_id": "whatever",
  "repos": ["repo1", "repo2"],
  "organizations": ["org1", "org2"],
  "followers_count": 82,
  "registered": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "quota_used_bytes": 55720 (only returned to users with Admin permission)
}
```

Security: Authenticated user is required

Get Organization

```
GET /orgs/:organization
```

Get information about a specified organization.
"type" inside the "members" list is available only to organization admins
"teams" list is available only to Premium organization admins

```
Status: 200 OK
{
    "name":"jfrog",
    "repos": ["repo1", "repo2"],
    "followers_count":0,
    "registered":"ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
    "owner":"joebloggs",
    "full_name":"JFrog",
    "members":
    [
        {"name": "user1", "type": "owner"},
        {"name": "user2", "type": "member"},
        {"name": "user3", "type": "admin"}
    ],
    "teams":["team1","team2"]
    "quota_used_bytes": 133410 (only returned to users with Admin permission)
}
```

Security: Authenticated user is required

Get Followers

```
GET /users/:user/followers[?start_pos=50]
```

Get followers of the specified repository owner

```
Status: 200 OK
[
  {"name": "user1"},
  {"name": "user2"}
]
```

Security: Authenticated user is required

User Search

```
GET /search/users?name=:name
```

Search for a user. Returns an array of results, where elements are similar to the result of getting a single user.

Security: Authenticated user is required

Subjects

Re-generate Subject URL Signing Key

```
POST /subjects/:subject/keypair
```

This resource is only available to Bintray Premium users.
For organization, caller must be an admin of the organization.
Re-generates Subject key for URL Signing.
Note: regenerating the URL signing key will revoke all active signed URLs.

```
Status: 201 Created
{
  "message": "success"
}
```

Security: Authenticated user with 'admin' permission.

Teams

This resource is only available to Bintray Premium users.
For organization, caller must be an admin of the organization.

Get Teams

```
GET /orgs/:org/teams
GET /users/:user/teams
```

Get a list of teams associated with an organization or a user

```
Status: 200 OK
{
  "teams":["team1", "team2", "team3"]
}
```

Security: Authenticated user with 'admin' permission.

Get Team

```
GET /orgs/:org/teams/:team
GET /users/:user/teams/:team
```

Get details of a team associated with an organization or a user

```
Status: 200 OK
{
  "name": "team1",
  "members":["user1", "user2", "user3"],
  "permitted_repos":{"repo1": "admin", "repo2": "publish"}}
}
```

Security: Authenticated user with 'admin' permission.

Create Team

```
POST /orgs/:org/teams
POST /users/:user/teams
```

Create a new team for an organization or a user

```
{
  "name":"team2",
  "members":["user1", "user2", "user3"]
  "allow_repo_creation": true
  "business_unit": "businessUnit1"     (only for Enterprise Account)
}
```

```
Status: 201 Created
{
  "name":"team2",
  "owner":"user4",
  "members":["user1", "user2", "user3"],
  "permitted_repo":[]
  "allow_repo_creation": true
  "business_unit": "businessUnit1"
}
```

"allow_repo_creation": team members are allowed to create and update repositories.
"business_unit": a default business unit can be associated to a team and will be the default business unit for all repositories that are created by this team. A business unit can only be associated with a team if its members are allowed to create repositories.

Security: Authenticated user with 'admin' permission.

Update Team

```
PATCH /orgs/:org/teams/:team
PATCH /users/:user/teams/:team
```

Update a team associated with an organization or a user

```
{
  "members":["user1", "user2"]
  "allow_repo_creation": true
  "business_unit": "businessUnit1"      (only for Enterprise Account)
}
```

```
Status: 200 OK
{
  "name":"team1",
  "owner":"user4",
  "members":["user1", "user2"],
  "permitted_repos":[]
  "allow_repo_creation": true
  "business_unit": "businessUnit1" ()
}
```

Security: Authenticated user with 'admin' permission.

Delete Team

```
DELETE /orgs/:org/teams/:team
DELETE /users/:user/teams/:team
```

Delete a team associated with an organization or a user

```
Status: 200 OK
{"message": "success"}
```

Security: Authenticated user with 'admin' permission.

Get All Team Permissions

```
GET /repos/:subject/:repo/permissions
```

Get the permissions defined for teams on the specified repository

```
Status: 200 OK
{
  ["team": "team1", "permission": "read"],
  ["team": "team2", "permission": "publish"]
}
```

Security: Authenticated user with 'admin' permission.

Get Team Permission

```
GET /repos/:subject/:repo/permissions/:team
```

Get the permissions defined for a team on the specified repository

```
Status: 200 OK
{
  "team":"team1",
  "permission":"publish"
}
```

Security: Authenticated user with 'admin' permission.

Set Team Permission

```
PUT /repos/:subject/:repo/permissions
```

Set the permissions defined for a team on the specified repository

```
{
  "team":"team1",
  "permission":"read"
}
```

```
Status: 200 OK
{"message": "success"}
```

Security: Authenticated user with 'admin' permission.

Delete Team Permission

```
DELETE /repos/:subject/:repo/permissions/:team
```

Delete the permission defined for a team on the specified repository

```
Status: 200 OK
{"message": "success"}
```

Security: Authenticated user with 'admin' permission.

Entitlements

This resource is only available to Bintray Pro and Enterprise users.

Get Access Keys

```
GET /orgs/:org/access_keys
GET /users/:user/access_keys
```

Get a list of access keys associated with an organization or a user

```
Status: 200 OK
{
  "access_keys":["key1", "key2", "key3"]
}
```

Security: Authenticated user with 'admin' permission.

Get Access Key

```
GET /orgs/:org/access_keys/:access_key_id
GET /users/:user/access_keys/:access_key_id
```

Get an access key associated with an organization or a user, by its id.

```
Status: 200 OK
{
  "id": "key1",
  "username": "key1@:subject",
  "expiry": 7956915742000,
  "existence_check": {
    "url": "http://callbacks.myci.org/username=:username,password=:password",
    "cache_for_secs": 60
  },
  "white_cidrs": ["127.0.0.1/22", "193.5.0.1/92"],
  "black_cidrs": ["197.4.0.1/4", "137.0.6.1/78"]
}
```

Security: Authenticated user with 'admin' permission.

Create Access Key

```
POST /orgs/:org/access_keys
POST /users/:user/access_keys
```

Create a new access key identified by an access key `id`, for an organization or a user.
An access key `password` will be auto-generated if not specified.

An optional expiry can be specified, after which the access key will be automatically revoked. Expiry value is in Unix epoch time in milliseconds.
Optionally, provide an existence check directive to verify whether the source identity of the access keys still exists.
Existence check uses a callback URL, optionally using the access key `username` and `password` as tokens. Only when a 404 is returned by the callback URL the access key will be automatically removed.
Callback results are cached for the specified period. The minimum value of `cache_for_secs` is 60 seconds.
Another option is to provide white and/or black CIDRs. Specifying white CIDRs will allow access only for those IPs that exist in that address range. Black CIDRs will block access for all IPs that exist in the specified range.
You can set api_only to false to allow access keys access to Bintray UI as well as to the API. Default value is true.

```
{
  "id": "key1",
  "expiry": 7956915742000,
  "existence_check": {
    "url": "http://callbacks.myci.org/username=:username,password=:password",
    "cache_for_secs": 60
  },
  "white_cidrs": ["127.0.0.1/22", "193.5.0.1/92"],
  "black_cidrs": ["197.4.0.1/4", "137.0.6.1/78"],
  "api_only": false
}
```

```
Status: 201 Created
{
  "username": "key1@:org",
  "password": "8fdf84d2a814783f0fc2ce869b5e7f6ce9f286a0"
}
```

Security: Authenticated user with 'admin' permission.

Delete Access Key

```
DELETE /orgs/:org/access_keys/:access_key_id
DELETE /users/:user/access_keys/:access_key_id
```

Delete an access key associated with an organization or a user.

```
Status: 200 OK
```

```
{"message": "success"}
```

Security: Authenticated user with 'admin' permission.

Update Access Key

```
PATCH /orgs/:org/access_keys/:access_key_id
PATCH /users/:user/access_keys/:access_key_id
```

Update an existing access key identified by an access key `id`, for an organization or a user.

`password`, `expiry`, `existence check` (`url` and `cache_for_secs`), `white_cidrs` and `black_cidrs` can be modified using this PATCH command. Only attributes specified in the command are affected. An access key `password` will be auto-generated if specified as empty string.

```
{
  "expiry": 7956915742000,
  "existence_check": {
    "url": "http://callbacks.myci.org/username=:username,password=:password",
    "cache_for_secs": 60
  },
  "white_cidrs": ["127.0.0.1/22", "193.5.0.1/92"],
  "black_cidrs": ["197.4.0.1/4", "137.0.6.1/78"]
}
```

```
Status: 200 OK
{
 "username": "key1@:org",
  "password": "8fdf84d2a814783f0fc2ce869b5e7f6ce9f286a0"
    "expiry": 7956915742000,
    "white_cidrs": [
        "127.0.0.1/22",
        "193.5.0.1/22"
    ],
    "black_cidrs": [
        "197.4.0.1/4",
        "137.0.6.1/22"
    ],
    "api_only": true,
    "existence_check": {
        "url": "http://callbacks.myci.org/username=:username,password=:password",
        "cache_for_secs": 70
    }
}
```

Security: Authenticated user with 'admin' permission.

Get Entitlements

```
GET /products/:subject/:product/entitlements   (only for Enterprise Account, if defi
GET /repos/:subject/:repo/entitlements
GET /packages/:subject/:repo/:package/entitlements
GET /packages/:subject/:repo/:package/versions/:version/entitlements
```

Get the entitlements defined on the specified product, repository, package or version.

```
Status: 200 OK
{
  ["id": "entitlement1"],
  ["id": "entitlement2"]
}
```

Security: Authenticated user with 'admin' permission.

Get Entitlement

```
GET /products/:subject/:product/entitlements/:entitlement_id   (only for Enterprise
GET /repos/:subject/:repo/entitlements/:entitlement_id
GET /packages/:subject/:repo/:package/entitlements/:entitlement_id
GET /packages/:subject/:repo/:package/versions/:version/entitlements/:entitlement_id
```

Get an entitlement by its id and scope. Scope can be a product, a repository, a package or a version.

```
Status: 200 OK
{
  "id": "7f8d57b16c1046e38062ea3db91838ff77758eca",
  "access": "rw",
  "download_keys": ["key1","key2"],
```

```
  "path": "a/b/c",
  "tags": ["tag1", "tag2"]
}
```

Security: Authenticated user with 'admin' permission.

Create Entitlement

```
POST /products/:subject/:product/entitlements   (only for Enterprise Account, if def
POST /repos/:subject/:repo/entitlements
POST /packages/:subject/:repo/:package/entitlements
POST /packages/:subject/:repo/:package/versions/:version/entitlements
```

Create an entitlement on the specified scope. Scope can be a product, a repository with an optional path, a package or a version.
When specifying an optional path value for repository scope, path needs to be relative and refer to a directory or a file in the repository.
Access mode can be either `rw` (read-write: implies download, upload and delete) or `r` (read: implies download).
When specifying a scope with product, access mode can only be `r` (read: implies download). Tags can be added for search purposes.
An entitlement `id` will be auto-generated if not specified.

```
{
  "access": "rw",
  "access_keys": ["key1","key2"],
  "path": "a/b/c",
  "tags": ["tag1", "tag2"]
}
```

```
Status: 201 Created
{
  "id": "7f8d57b16c1046e38062ea3db91838ff77758eca",
  "path": "a/b/c",
  "access": "rw",
  "access_keys": ["key1","key2"],
  "tags": ["tag1", "tag2"]
}
```

Security: Authenticated user with 'admin' permission.

Delete Entitlement

```
DELETE /products/:subject/:product/entitlements/:entitlement_id   (only for Enterpri
DELETE /repos/:subject/:repo/entitlements/:entitlement_id
DELETE /packages/:subject/:repo/:package/entitlements/:entitlement_id
DELETE /packages/:subject/:repo/:package/versions/:version/entitlements/:entitlement
```

Delete an entitlement by its id and scope. Scope can be a product, a repository, a package or a version.

```
Status: 200 OK
{"message": "success"}
```

Security: Authenticated user with 'admin' permission.

Update Entitlement

```
PATCH /products/:subject/:product/entitlements/:entitlement_id   (only for Enterpris
PATCH /repos/:subject/:repo/entitlements/:entitlement_id
PATCH /packages/:subject/:repo/:package/entitlements/:entitlement_id
PATCH /packages/:subject/:repo/:package/versions/:version/entitlements/:entitlement_
```

Update the information of the specified entitlement of a specified scope. Scope can be a product, a repository with an optional path, a package or a version.
'access', 'access_keys', 'access_keys_add', 'access_keys_remove' and 'tags' can be modified using this PATCH command.
'access_keys': a list of keys to replace the current list of access keys.
'access_keys_add': a list of keys to append to current list of access keys.
'access_keys_remove': a list of keys to remove from the current list of access keys.
'tags': a list of tags to replace the current list of tags.
'tags_add': a list of tags to append to current list of tags.
'tags_remove': a list of tags to remove from the current list of tags.

```
{
  "access": "rw",
  "access_keys": ["key1","key2"],
  "tags": ["tag1", "tag2"]
}
```

```
Status: 200 OK
{
  "id": "7f8d57b16c1046e38062ea3db91838ff77758eca",
  "path": "a/b/c"
  "access": "rw",
  "access_keys": ["key1","key2"],
  "tags": ["tag1", "tag2"]
}
```

Security: Authenticated user with 'admin' permission.

Entitlement Search By Access Key

```
GET /search/entitlements?access_key=:access_key&scope=:subject[/:repo][/:package][/:
```

Search for entitlements for a specific access key in the specified scope.
The minimal scope is a subject. You can optionally add a sub-scope of product, repo, package and version.
If deep equals true (default is false), will return all entitlements under the given scope, for example, if scope is repository, existing package and version entitlements under the given repository will be returned.

For example:

```
GET https://api.bintray.com/search/entitlements?access_key=key1&scope=jfrog/test-repo
```

```
Status: 200 OK
[
   {
     "id": "entitlement-id",
     "repo": "/jfrog/test-repo",
     "path": "com/jfrog/bintray/0.1.0",
     "access": "r",
     "access_keys": ["key1"]
   }
]
```

Security: Authenticated user with 'admin' permission.

Entitlement Search By Tag

```
GET /search/entitlements?tag=:tag&scope=:subject[/:repo][/:package][/:version][&prod
```

Search for entitlements for a specific tag in the specified scope.
The minimal scope is a subject. You can optionally add a sub-scope of product, repo, package and version.
If deep equals true (default is false), will return all entitlements under the given scope, for example, if scope is repository, existing package and version entitlements under the given repository will be returned.

For example:

```
GET https://api.bintray.com/search/entitlements?tag=tag1&scope=jfrog/test-repo
```

```
Status: 200 OK
[
   {
     "id": "entitlement-id",
     "repo": "/jfrog/test-repo",
     "path": "com/jfrog/bintray/0.1.0",
     "access": "r",
     "access_keys": ["key1"],
     "tags": ["tag1"]
   }
]
```

Security: Authenticated user with 'admin' permission.

Statistics & Usage Report

All statistics APIs are using date and time range. For the past twenty four hours, time resolution is minutes. If the provided date is earlier than the past twenty four hours, the provided time will be set to beginning of the provided date.
Statistics can only be retrieved for Premium accounts, and require a publish permission or higher.

Get Daily Downloads

```
POST /packages/:subject/:repo/:package/stats/time_range_downloads
POST /packages/:subject/:repo/:package/versions/:version/stats/time_range_downloads
```

Get number of downloads per day, for the passed time range, per package or per version.

```
{
  "from": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "to": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)"
}
```

```
Status: 200 OK
{
  "from": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "to": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
For package
  "records":[{"date":"yyyy/MM/dd","downloads":[{"version":"version1","count":4}]},
             {"date":"yyyy/MM/dd","downloads":[{"version":"version1","count":1},{"version":"v
For version
  "records":[{"date":"yyyy/MM/dd","downloads":[{"version":"version1","count":4}]},
             {"date":"yyyy/MM/dd","downloads":[{"version":"version1","count":1}]}]
}
```

Security: Authenticated user with 'publish' permission for private repositories, or package read/write entitlement.

Get Total Downloads

```
POST /packages/:subject/:repo/:package/stats/total_downloads
POST /packages/:subject/:repo/:package/versions/:version/stats/total_downloads
```

Get total number of downloads, for the passed time range, per package or per version.

```
{
  "from": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "to": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)"
}
```

```
Status: 200 OK
{
  "from": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "to": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
For package
  "records":[{"version":"version1","count":1},{"version":"version2","count":10}]
For version
  "records":[{"version":"version1","count":1}]
}
```

Security: Authenticated user with 'publish' permission for private repositories, or package read/write entitlement.

Get Downloads by Country

```
POST /packages/:subject/:repo/:package/stats/country_downloads
POST /packages/:subject/:repo/:package/versions/:version/stats/country_downloads
```

Get number of downloads by country, for the passed time range, per package or per version.

```
{
  "from": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "to": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)"
}
```

```
Status: 200 OK
{
```

```
  "from": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "to": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "records":[{"country":"country1","count":1}]
}
```

Security: Authenticated user with 'publish' permission for private repositories, or package read/write entitlement.

Get Usage Report For Subject

```
POST /usage/:subject
```

This resource is only available for Bintray Premium accounts.

Get monthly download and storage usage report, according to the specified date range for a subject.

```
{
  "from": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "to": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)"
}
```

```
Status: 200 OK
[
  {
    "from": "2016-04-01T00:00:00.000Z",
    "to": "2016-05-01T00:00:00.000Z",
    "partial_period": false,
    "download_bytes": 190023453467,
    "storage_bytes": 30788543
  },
  {
    "from": "2016-05-01T00:00:00.000Z",
    "to": "2016-05-20T00:00:00.000Z",
    "partial_period": true,
    "download_bytes": 126028851487,
    "storage_bytes": 45762911
  }
]
```

Security: Authenticated user with 'admin' permission.

Get Usage Report For Repository

```
POST /usage/:subject/:repo
```

This resource is only available for Bintray Premium accounts.

Get monthly download and storage usage report, according to the specified date range for a specific subject repository.

```
{
  "from": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "to": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)"
}
```

```
Status: 200 OK
[
  {
    "business_unit": "businessUnit1",
    "from": "2016-12-01T00:00:00.000Z",
    "to": "2017-01-01T00:00:00.000Z",
    "partial_period": false,
    "download_bytes": 17351146,
    "download_percentage": 8.16136697,
    "storage_bytes": 640048090,
    "storage_percentage": 29.17027826
  },
  {
    "business_unit": "businessUnit1",
    "from": "2017-01-01T00:00:00.000Z",
    "to": "2017-01-20T16:00:32.854Z",
    "partial_period": true,
    "download_bytes": 106386794,
    "download_percentage": 50.04059483,
    "storage_bytes": 533694374,
    "storage_percentage": 24.32319327
  }
]
```

Security: Authenticated user with 'admin' permission.

Get Usage Report For Package

```
POST /usage/package_usage/:subject/:repo/[:package][?start_pos=50]
```

This resource is only available for Bintray Premium accounts.

Get current storage usage report. Report can be requested for the specified repository, optionally for a specific package.

```
Status: 200 OK
[
    {
        "package": "pack1",
        "storage_bytes": 98082192,
        "file_count": 59
    },
    {
        "package": "pack3",
        "storage_bytes": 842052350,
        "file_count": 300
    }
]
```

Security: Authenticated user with 'admin' permission for repo, or 'publish' permission for specific package.

Get Usage Report Grouped By Business Unit

```
POST /usage/business_unit_usage/:subject[/:business_unit]
```

This resource is only available for Bintray Enterprise accounts.

Get monthly download and storage usage report, according to the specified date range and grouped by business unit. Report can be requested for a subject or for a specific subject business unit.

```
{
  "from": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "to": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)"
}
```

```
Status: 200 OK
[
  {
    "business_unit": "businessUnit1",
    "from": "2016-12-01T00:00:00.000Z",
    "to": "2017-01-01T00:00:00.000Z",
    "partial_period": false,
    "download_bytes": 17351146,
    "download_percentage": 8.16136697,
    "storage_bytes": 640048090,
    "storage_percentage": 29.17027826
  },
  {
    "business_unit": "businessUnit1",
    "from": "2017-01-01T00:00:00.000Z",
    "to": "2017-01-20T16:00:32.854Z",
    "partial_period": true,
    "download_bytes": 106386794,
    "download_percentage": 50.04059483,
    "storage_bytes": 533694374,
    "storage_percentage": 24.32319327
  },
  {
    "business_unit": "businessUnit2",
    "from": "2016-12-01T00:00:00.000Z",
    "to": "2017-01-01T00:00:00.000Z",
    "partial_period": false,
    "download_bytes": 88863038,
    "download_percentage": 41.7980382,
    "storage_bytes": 438190535,
    "storage_percentage": 19.97059289
  },
  {
    "business_unit": "businessUnit2",
    "from": "2017-01-01T00:00:00.000Z",
    "to": "2017-01-20T16:00:32.854Z",
    "partial_period": true,
    "download_bytes": 0,
    "download_percentage": 0,
```

```
    "storage_bytes": 438190535,
    "storage_percentage": 19.97059289
  }
]
```

Security: Authenticated user with 'admin' permission.

Usage Thresholds

This resource is only available for Bintray Enterprise accounts.

There are three types of usage threshold that can be set for organizations / repositories / business units:
Maximum monthly storage
Monthly download volume
Daily download volume
Once a threshold is exceeded, a notification will be sent through the firehose stream.
Email notifications can be sent to organization admins and to additional specified recipients.

Get Usage Threshold

```
GET /usage_threshold/organization/:org
GET /usage_threshold/repo/:org/:repo
GET /usage_threshold/business_unit/:org/:business_unit
```

```
Status: 200 OK
{
    "monthly_storage_bytes": 10000000,
    "monthly_download_bytes": 0,
    "daily_download_bytes": 10000,
    "alert_to_emails": [
      "recipient1@email.com",
      "recipient2@email.com"
    ],
    "alert_to_admins": true
}
```

Security: Authenticated user with organization ‘admin’ permission.

Create Usage Threshold

```
POST /usage_threshold/organization/:org
POST /usage_threshold/repo/:org/:repo
POST /usage_threshold/business_unit/:org/:business_unit
```

At least one threshold type must be specified.

```
Status: 201 CREATED
{
    "monthly_storage_bytes": 10000000,
    "monthly_download_bytes": 10000000,
    "daily_download_bytes": 10000,
    "alert_to_emails": [
      "recipient1@email.com",
      "recipient2@email.com"
    ],
    "alert_to_admins": true (default)/false
}
```

Body: JSON of the created usage threshold

Security: Authenticated user with organization 'admin' permission.

Update Usage Threshold

```
PATCH /usage_threshold/organization/:org
PATCH /usage_threshold/repo/:org/:repo
PATCH /usage_threshold/business_unit/:org/:business_unit
```

Set a threshold to 0 in order to disable notifications for a specific event type.

```
Status: 200 OK
{
    "monthly_storage_bytes": 10000000,
    "monthly_download_bytes": 0,
```

```
    "daily_download_bytes": 10000,
    "alert_to_emails": [
      "recipient1@email.com",
      "recipient2@email.com"
    ],
    "alert_to_admins": true
}
```

Security: Authenticated user with organization 'admin' permission.

Delete Usage Thresholds

```
DELETE /usage_threshold/organization/:org
DELETE /usage_threshold/repo/:org/:repo
DELETE /usage_threshold/business_unit/:org/:business_unit
```

```
Status: 200 OK
```

Security: Authenticated user with organization 'admin' permission.

Licenses

This resource is only available to Bintray Premium users.

Get Proprietary Licenses

```
GET /orgs/:org/licenses
GET /users/:user/licenses
```

Get a list of custom, proprietary licenses associated with an organization or a user.
For organization, caller must have publishing permission on at least one repository.

```
Status: 200 OK
[
  {
    "name": "custom-eula",
    "description":"EULA for product x",
    "url":"https://my-commercial-license.com"
  }
]
```

Security: Authenticated user with 'publish' permission or higher is required. Requires 'admin' permission for user

Create Proprietary License

```
POST /orgs/:org/licenses
POST /users/:user/licenses
```

Create a license associated with an organization or a user.
For organization, caller must be an admin of the organization.

```
[
  {
    "name": "license1",
    "description":"license-1",
    "url":"https://licenses.com"
  }
]
```

```
Status: 201 Created
{License get JSON response}
```

Security: Authenticated user with 'admin' permission.

Update Proprietary License

```
PATCH /orgs/:org/licenses/:custom_license_name
PATCH /users/:user/licenses/:custom_license_name
```

Update a license associated with an organization or a user.
For organization, caller must be an admin of the organization.

```
[
  {
    "description":"license-1",
    "url":"https://licenses.com"
  }
]
```

```
Status: 200 OK
{License get JSON response}
```

Security: Authenticated user with 'admin' permission.

Delete Proprietary License

```
DELETE /orgs/:org/licenses/:custom_license_name
DELETE /users/:user/licenses/:custom_license_name
```

Delete a license associated with an organization or a user.
For organization, caller must be an admin of the organization.

```
Status: 200 OK
{"message": "success"}
```

Security: Authenticated user with 'admin' permission.

Get OSS Licenses

```
GET /licenses/oss_licenses
```

Returns a list of all the OSS licenses. This resource can be consumed by both authenticated and anonymous clients.

```
[
  {
    "name":"Apache-1.0",
    "longname":"The Apache Software License, Version 1.0",
    "url":"http://apache.org/licenses/LICENSE-1.0"
  }
]
```

```
Status: 200 OK
```

Logs

This resource is only available to Bintray Premium users.

List Package Download Log Files

```
GET /packages/:subject/:repo/:package/logs
```

Retrieve a list of available download log files for a package

```
Status: 200 OK
[
  {
    "name": "downloads-05-11-2013.log.gz",
    "size": 209,
    "update": "2013-11-05T12:55:00Z",
  },
  …
]
```

Security: Authenticated user with 'publish' permission, or package read/write entitlement.

Download Package Download Log File

```
GET /packages/:subject/:repo/:package/logs/:log_name
```

Download the package download log file specified by log_name

```
Status: 200 OK
```

Security: Authenticated user with 'publish' permission, or package read/write entitlement.

Stream API (Events Firehose)

```
GET /stream/:subject
```

Get a stream of events generated by activity for the specified subject.
Currently, the following event types are supported: `download`, `upload`, `delete`, `login_success`, and `login_failure`. The response body is standard http with chunked transfer encoding (`Transfer-Encoding: chunked`).
It consists of newline "\r\n" delimited messages, where each message can be either a single line JSON String representing an event or an empty string. Empty strings are used for keep-alive. Empty keep-alive message are sent every 30 seconds. Clients are expected to disconnect/reconnect if they did not receive a keep-alive message for 60 seconds.

`X-Bintray-Stream-Reconnect-Id`: The response will include the header `X-Bintray-Stream-Reconnect-Id` with a value. This value can be used for re-establishinghu connection to the same event stream. Disconnected clients should send the reconnect id using the same header (`X-Bintray-Stream-Reconnect-Id`). Bintray will reconnect the client to the existing stream and will stream back any events generated in the past 120 seconds since the stream has been disconnected. This allows clients enough time to reestablish/refresh connections without losing events.
It is expected from clients not to share the reconnect id across different connections.

Events compression can be enabled by supplying the `Accept-Encoding: gzip` request header. In this case, response compression is indicated with the `Content-Encoding: gzip` response header.

Limitations: At present, there is a limit of 3 concurrent connections for a given subject.
Pleasse note that there may be a wait time of up to 120 seconds between reconnection attempts when a reconnect id is not supplied. E.g.: If 3 clients have been started and one of them is disconnecting+reconnecting without supplying a reconnect id, it may need to wait for 120 seconds until a successful connection can be established, since events are saved on the old connection for that period of time.
In the case of too many connections a 429 (TOO MANY REQUESTS) response error code will be returned to connecting clients.

Security: Authenticated subject admin.

Example:

```
HTTP/1.1 200 OK
{"type":"login_success","time":"2016-12-05T06:32:42.987Z","subject":"user@myorg","ip_address"
{"type":"login_failure","time":"2016-12-05T06:33:01.011Z","subject":"user@myorg","ip_address"
{"type":"upload","path":"/myorg/mavenp/app.jar","subject":"user","time":"2016-12-05T06:33:49.
{"type":"download","path":"/myorg/mavenp/app.jar","subject":"user","time":"2016-12-05T06:34:0
{"type":"delete","path":"/myorg/mavenp/app.jar","subject":"user","time":"2016-12-05T06:34:06.
```

Webhooks

Get Webhooks

```
GET /webhooks/:subject[/:repo]
```

Get all the webhooks registered for the specified subject, optionally for a specific repository.

failure_count is the number of times a callback has failed. A callback will be auto-deactivated after 7 subsequent failures. A successful callback resets the count.

```
Status: 200 OK
[
  {
    "package": "my-package",
    "url": "http://callbacks.myci.org/%r-%p-build",
    "failure_count": "3",
  },
  …
```

```
]
```

Security: Authenticated user with 'read' permission, or repository read/write entitlement.

Register a Webhook

```
POST /webhooks/:subject/:repo/:package
```

Register a webhook for receiving notifications on a new package release. By default a user can register up to 10 webhook callbacks. The callback URL may contain the %r and %p tokens for repo and package name, respectively. method is the callback request method: can be in post, put or get. If not specified, post is used.

```
{
  "url": "http://callbacks.myci.org/%r-%p-build",
  "method": "post"
}
```

```
Status: 201 Created
X-Bintray-WebHookLimit-Limit: 10
X-Bintray-WebHookLimit-Remaining: 2
```

Security: Authenticated user with 'publish' permission, or package read/write entitlement.

Test a Webhook

```
POST /webhooks/:subject/:repo/:package/:version
```

Test a webhook callback for the specified package release. A webhook post request is authenticated with an HMAC-SHA256 authentication header of the package name keyed by the registering subject's API key, and base64-encoded.

```
{
  "url": "http://callbacks.myci.org/%r-%p-build",
  "method": "GET"
}
```

```
Status: 200 OK
"X-Bintray-Hook-Hmac": "Base64 HMAC-SHA256 of the package name keyed by the API key."
{
  "package": "my-package",
  "version": "1.2.1",
  "released": "ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)",
  "release_notes": "This is a test" (TBD)
}
```

Security: Authenticated user with 'publish' permission, or package read/write entitlement.

Delete a Webhook

```
DELETE /webhooks/:subject/:repo/:package
```

Delete a user's webhook associated with the specified package.

```
Status: 200 OK
{
  "message": "success"
}
```

Security: Authenticated user with 'publish' permission, or package read/write entitlement.

Docker

The following Docker terminology applies to Bintray's terminology:
Docker Registry is effectively a Bintray Repository, capable of hosting an unlimited number of Docker repositories. A Bintray repository has a registry domain name of: `:subject-docker-:repo.bintray.com`.
Docker Repository is a Bintray Package, where a Docker namespace is translated to a prefix of a Bintray package name.
Docker Tag is translated to a Bintray Version.

Delete Docker Repository

Delete a Docker repository can be done via one of the following options:

1. Using Bintray's Delete Package rest api
2. Using Docker's syntax:

```
DELETE https://:dockerRegistry/v1/repositories/:namespace/:dockerRepository
```

For example:

```
DELETE https://jfrog-docker-registry.bintray.io/v1/repositories/bintray/dockerrepo
```

Security: Authenticated user with 'admin' permission.

Delete Docker Tag

Delete a Docker tag can be done via one of the following options:

1. Using Bintray's Delete Version rest api
2. Using Docker's syntax:

```
DELETE https://:dockerRegistry/v1/repositories/:namespace/:dockerRepository/tags/:tagName
```

For example:

```
DELETE https://jfrog-docker-registry.bintray.io/v1/repositories/bintray/dockerrepo/tags/1
```

Security: Authenticated user with 'publish' permission, or repository read/write entitlement.

Available licenses:
AFL-2.1, AFL-3.0, AGPL-V3, Apache-1.0, Apache-1.1, Apache-2.0, APL-1.0, APSL-2.0, Artistic-License-2.0, Attribution, Bouncy-Castle, BSD, BSD 2-Clause, BSD 3-Clause, BSL-1.0, CA-TOSL-1.1, CC0-1.0, CDDL-1.0, Codehaus, CPAL-1.0, CPL-1.0, CPOL-1.02, CUAOFFICE-1.0, Day, Day-Addendum, ECL2, Eiffel-2.0, Entessa-1.0, EPL-1.0, EPL-2.0, EUDATAGRID, EUPL-1.1, EUPL-1.2, Fair, Facebook-Platform, Frameworx-1.0, Go, GPL-2.0, GPL-2.0+CE, GPL-3.0, Historical, HSQLDB, IBMPL-1.0, IJG, ImageMagick, IPAFont-1.0, ISC, IU-Extreme-1.1.1, JA-SIG, JSON, JTidy, LGPL-2.0, LGPL-2.1, LGPL-3.0, Libpng, LPPL-1.0, Lucent-1.02, MirOS, MIT, Motosoto-0.9.1, Mozilla-1.1, MPL-2.0, MS-PL, MS-RL, Multics, NASA-1.3, NAUMEN, NCSA, Nethack, Nokia-1.0a, NOSL-3.0, NTP, NUnit-2.6.3, NUnit-Test-Adapter-2.6.3, OCLC-2.0, Openfont-1.1, Opengroup, OpenSSL, OSL-3.0, PHP-3.0, PostgreSQL, Public Domain, Public Domain - SUN, PythonPL, PythonSoftFoundation, QTPL-1.0, Real-1.0, RicohPL, RPL-1.5, Scala, SimPL-2.0, Sleepycat, SUNPublic-1.0, Sybase-1.0, TMate, Unicode-DFS-2015, Unlicense, UoI-NCSA, UPL-1.0, VIM License, VovidaPL-1.0, W3C, WTFPL, wxWindows, Xnet, ZLIB, ZPL-2.0

Last updated 2018-06-25 09:37:24 +00:00