CloudBlue
https://cloudblue.com

API Compatibility and Versioning Policy

This article has been generated from the online version of the documentation and might be out of date. Please, make sure to always refer to the online version of the documentation for the up-to-date information.

Auto-generated at December 5, 2023

In this document, version refers to a specific representation of a CloudBlue Connect REST API data structure definitions and functionality. Compatibility refers to the ability of a client to execute against a specific version of the API. The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119.

Versioning

REST API of the CloudBlue Connect are versioned using major version, which is defined by the URL path `/public/{major}` of the API Endpoint. Each **{major}** version is a string in the format of “v1”, “v2”, etc.

Currently, most collections have a single major version (“v1”). As additional versions of APIs are added, clients request a specific version of a representation using specific URL path. If the service supports the exact version requested, the exact version of the resource is always returned.

Clients that request a deprecated version receive the most recent supported version. Please watch the Release Notes for announcements regarding implementation of this behavior.

Backward Compatibility

An API is Backward Compatible if a *supported client* written against a version of that API will continue to work the same way, without modification, against future versions of the API. Only cli*ents that follow recommendations* outlined in this policy are supported by the CloudBlue Connect.

Resources in the CloudBlue Connect REST API are guaranteed to be backward compatible within a major version. Backward compatibility is guaranteed through the use of stable resource and URI representation principle.

Stable URIs

If a resource exists at a particular URI, that same resource will continue to exist with the same meaning in future versions. This means:

1. The meaning of HTTP response codes MUST be trusted. If a URI that used to return 200 now returns a 404, you know it is because the resource cannot be found, not because the resource has moved to another location.
2. A resource or a collection MAY support additional query parameters in future versions but they WILL NOT be mandatory. The absence of a value or a default value (as appropriate) will maintain prior behaviour.
3. If a resource or a collection accepts a representation (e.g. via POST or PUT), it will continue to accept the same representation in future versions. Any additional properties that are recognised in a resource WILL NOT be mandatory, and the default value assumed in their absence will be chosen to maintain the previous meaning of the resource.
4. A resource or a collection MAY be modified to return a “redirection” response code (e.g. 301, 302) instead of directly returning the resource. Clients MUST handle HTTP-level redirects, and respect HTTP headers (e.g. `Location`).

Stable Represenatations

If the resource at a URI is documented as being available in a specific media type, e.g. via the `Content-Type` header when submitting resource and `Accept` header when requesting a resource or a collection, that media type will be maintained. If a resource returns a default media type in the absence of content negotiation, that default will be maintained.

Resources with a media type of `application/json` have additional stability guarantees:

1. A property defined in a resource representation MUST constant in name and type (i.e. string, number, array or object).
2. The definition of a property MUST remain consistent and unchanging within a major version.
3. New properties MAY be defined on a resource represenation within the same major version, but a new property MUST NOT modify the meaning of any previously defined property.
4. If a property has a primitive type and the API documentation does not explicitly limit its possible values, clients MUST NOT assume the values are constrained to a particular set of possible responses.
5. New possible values of primitive types like enumerations MAY be added within a major version, though definition of already defined enumeration values MUST remain consistent and unchanging within a major version.
6. If a property of an object is not explicitly declared as mandatory in the API, clients MUST NOT assume it will be present.
7. If a property of an object is a URI, then the resource identified by that URI MUST maintain the same compatibility guarantee.

In the case where stable resource represebtation principle could not be followed, the CloudBlue Connect team will communicate the change and work with clients to migrate to the new version of the stable representation where possible.

Extensions Compatibility

Certain parts of the CloudBlue Connect API are able to be extended by Extensions written by 3rd party developers. Extensions of the platform MUST follow the compatibility guidelines.

However, since plugins MAY be enabled, disabled, upgraded or downgraded at any time without warning, clients MUST NOT assume that data provided by extensions will be present in a response, or that extension data provided in a mutating request will necessarily be saved.

Forward Compatibility

An API is Forward Compatible if a client written against a version of the API will also work the same way, without modification, against previous versions of the API. We make NO guarantees of Forward Compatibility in our REST APIs, but we provide the following non-normative guidelines about our approach to the forward compatibility:

> ●
> Postel’s Law (Robustness Principle)
>
> Be conservative in what you do, be liberal in what you accept from others.

Where possible, we follow the Robustness Principle above. This means that the API will determine what to do with a request based only on the parts of that it recognises.

1. Request query parameters that are not recognised by the server will be ignored where technically possible.
2. Expansion parameters that are not recognised by the server will be ignored where technically possible.
3. Properties of structured (i.e. JSON) data submitted via mutative requests that are not recognised by the server will be ignored where technically possible.
4. Data that is intended to be passed to a plugin extension that is not installed or is not active will be ignored where technically possible.

Data Minimisation

We strongly recommend clients to follow the Data Minimisation principle, i.e. making sure that data they consume and send to the API is

1. **adequate** – sufficient to properly fulfil your stated purpose
2. **relevant** – has a rational link to that purpose
3. limited to what is **necessary** – you do not hold more than you need for that purpose

You should work with only that much information, but no more.

Rate limits and paging

Resource and rate limits, and the default and maximum sizes of paged data ARE NOT considered part of the API and MAY change (possibly dynamically). If a page of a requested size could not be returned, a smaller amount of data MAY be returned and it is a responsibility fo a client to analyze the `Content-Range` header to understand if an additional page needs to be requested.

A platform MAY use HTTP error code `429 Too Many Requests` to indicate that too many requests were sent by a client in a given amount of time. A `Retry-After` header MAY be included to this response indicating how long to wait before making a new request. It is the responsibility of the client to read the road signs and obey the speed limit.

Deprecation

The CloudBlue Connect team makes every effort to inform clients of major version deprecation at least 3 months in advance. The CloudBlue Connect team provides migration guidance to assist clients to migrate to the latest version.

In the case where stable resource representation is no longer returned due to legal issue, critical bug, or critical security flaw, teh associated stable resource representation is deprecated as soon as possible.