shalako path may be clobbered by bind-mount e35366b on Jun 21, 2017

10 contributors

943 lines (709 sloc) 53.1 KB

Open Service Broker API v2.12

Table of Contents

API Overview

The Service Broker API defines an HTTP interface between the services marketplace of a platform and service brokers.

The service broker is the component of the service that implements the Service Broker API, for which a platform's marketplace is a client. Service brokers are responsible for advertising a catalog of service offerings and service plans to the marketplace, and acting on requests from the marketplace for provisioning, binding, unbinding, and deprovisioning.

In general, provisioning reserves a resource on a service; we call this reserved resource a service instance. What a service instance represents can vary by service. Examples include a single database on a multi-tenant server, a dedicated cluster, or an account on a web application.

What a binding represents MAY also vary by service. In general creation of a binding either generates credentials necessary for accessing the resource or provides the service instance with information for a configuration change.

A platform marketplace MAY expose services from one or many service brokers, and an individual service broker MAY support one or many platform marketplaces using different URL prefixes and credentials.

Notations and Terminology

Notational Conventions

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.

Terminology

This specification defines the following terms:

- *Platform*: The software that will manage the cloud environment into which Applications and Service Brokers are provisioned. Users will not directly provision Services from Service Brokers, rather they will ask the Platform (ie. their cloud provider) to manage Services and interact with the Service Brokers for them.
- *Service*: A managed software offering that can be used by an Application. Typically, Services will expose some API that can be invoked to perform some action. However, there can also be non-interactive Services that can perform the desired actions without direct prompting from the Application.
- *Service Broker*: Service Brokers manage the lifecycle of Services. Platforms interact with Service Brokers to provision, and manage, Service Instances and Service Bindings.
- *Service Instance*: An instantiation of a Service offering.
- *Service Binding*: The representation of an association between an Application and a Service Instance. Often, Service Bindings, will contain the credentials that the Application will use to communicate with the Service Instance.
- *Application*: The software that will make use of, be bound to, a Service Instance.

Changes

Change Policy

- Existing endpoints and fields MUST NOT be removed or renamed.
- New OPTIONAL endpoints, or new HTTP methods for existing endpoints, MAY be added to enable support for new features.
- New fields MAY be added to existing request/response messages. These fields MUST be OPTIONAL and SHOULD be ignored by clients and servers that do not understand them.

Changes Since v2.11

- Added `context` field to request body for provisioning a service instance (`PUT /v2/service_instances/:instance_id`) and updating a service instance (`PATCH /v2/service_instances/:instance_id`). Also added the profile.md file describing RECOMMENDED usage patterns for environment-specific extensions and variations. `context` will replace the `organization_guid` and `space_guid` request fields in future versions of the specification. In the interim, both SHOULD be used to ensure interoperability with old and new implementations.
- The specification now uses RFC2119 keywords to make it clearer when things are REQUIRED rather than OPTIONAL or RECOMMENDED.
- Request and response parameters of type string, if present, MUST be a non-empty string.
- Cleaned up text around status codes in responses, clarifying when they MUST be returned.
- Added clarity around the `app_guid` request field.

- Clarified the semantics of `plan_id` and `parameters` fields in the updating a service instance request body.
- Clarified the default value of the `plan_updateable` field.
- Clarified when `route_service_url` is REQUIRED and when brokers can return extra data in bindings.

For changes in older versions, see the release notes.

API Version Header

Requests from the platform to the service broker MUST contain a header that declares the version number of the Service Broker API that the marketplace will use:

`X-Broker-Api-Version: 2.12`

The version numbers are in the format `MAJOR.MINOR` using semantic versioning.

This header allows brokers to reject requests from marketplaces for versions they do not support. While minor API revisions will always be additive, it is possible that brokers depend on a feature from a newer version of the API that is supported by the platform. In this scenario the broker MAY reject the request with `412 Precondition Failed` and provide a message that informs the operator of the API version that is to be used instead.

Authentication

The marketplace MUST authenticate with the service broker using HTTP basic authentication (the `Authorization:` header) on every request. The broker is responsible for validating the username and password and returning a `401 Unauthorized` message if credentials are invalid. It is RECOMMENDED that brokers support secure communication from platform marketplaces over TLS.

Catalog Management

The first endpoint that a platform will interact with on the broker is the service catalog.

The platform marketplace fetches this endpoint from all brokers in order to present an aggregated user-facing catalog.

Warnings for broker authors:

- Be cautious removing services and plans from their catalogs, as platform marketplaces might have provisioned service instances of these plans. Consider your deprecation strategy.
- Do not change the ids of services and plans. This action is likely to be evaluated by a platform marketplace as a removal of one plan and addition of another. See above warning about removal of plans.

The following sections describe catalog requests and responses in the Service Broker API.

Request

Route

`GET /v2/catalog`

cURL

```
$ curl -H "X-Broker-API-Version: 2.12" http://username:password@broker-url/v2/catalog
```

Response

Status Code	Description
200 OK	MUST be returned upon successful processing of this request. The expected response body is below.

Body - Schema of Service Objects

CLI and web clients have different needs with regard to service and plan names. A CLI-friendly string is all lowercase, with no spaces. Keep it short -- imagine a user having to type it as an argument for a longer command. A web-friendly display name is camel-cased with spaces and punctuation supported.

Response field	Type	Description
services*	array-of-service-objects	Schema of service objects defined below. MAY be empty.

Service Objects

Response field	Type	Description
name*	string	A CLI-friendly name of the service. All lowercase, no spaces. This MUST be globally unique within a platform marketplace. MUST be a non-empty string.
id*	string	An identifier used to correlate this service in future requests to the broker. This MUST be globally unique within a platform marketplace. MUST be a non-empty string. Using a GUID is RECOMMENDED.
description*	string	A short description of the service. MUST be a non-empty string.
tags	array-of-strings	Tags provide a flexible mechanism to expose a classification, attribute, or base technology of a service, enabling equivalent services to be swapped out without changes to dependent logic in applications, buildpacks, or other services. E.g. mysql, relational, redis, key-value, caching, messaging, amqp.
requires	array-of-strings	A list of permissions that the user would have to give the service, if they provision it. The only permissions currently supported are `syslog_drain`, `route_forwarding` and `volume_mount`.
bindable*	boolean	Specifies whether service instances of the service can be bound to applications. This specifies the default for all plans of this service. Plans can override this field (see Plan Object).
metadata	JSON object	An opaque object of metadata for a service offering. Controller treats this as a blob. Note that there are conventions in existing brokers and controllers for fields that aid in the display of catalog data.
dashboard_client	object	Contains the data necessary to activate the Dashboard SSO feature for this service.
plan_updateable	boolean	Whether the service supports upgrade/downgrade for some plans. Please note that the misspelling of the attribute `plan_updatable` to `plan_updateable` was done by mistake. We have opted to keep that misspelling instead of fixing it and thus breaking backward compatibility. Defaults to false.
plans*	array-of-objects	A list of plans for this service, schema is defined below. MUST contain at least one plan.

Dashboard Client Object

Response field	Type	Description
id	string	The id of the Oauth client that the dashboard will use. If present, MUST be a non-empty string.
secret	string	A secret for the dashboard client. If present, MUST be a non-empty string.
redirect_uri	string	A URI for the service dashboard. Validated by the OAuth token server when the dashboard requests a token.

Plan Object

Response field	Type	Description
id*	string	An identifier used to correlate this plan in future requests to the broker. This MUST be globally unique within a platform marketplace. MUST be a non-empty string. Using a GUID is RECOMMENDED.
name*	string	The CLI-friendly name of the plan. MUST be unique within the service. All lowercase, no spaces. MUST be a non-empty string.
description*	string	A short description of the plan. MUST be a non-empty string.
metadata	JSON object	An opaque object of metadata for a service plan. Controller treats this as a blob. Note that there are conventions in existing brokers and controllers for fields that aid in the display of catalog data.
free	boolean	When false, service instances of this plan have a cost. The default is true.
bindable	boolean	Specifies whether service instances of the service plan can be bound to applications. This field is OPTIONAL. If specified, this takes precedence over the `bindable` attribute of the service. If not specified, the default is derived from the service.

* Fields with an asterisk are REQUIRED.

```
{
  "services": [{
    "name": "fake-service",
    "id": "acb56d7c-XXXX-XXXX-XXXX-feb140a59a66",
    "description": "fake service",
    "tags": ["no-sql", "relational"],
    "requires": ["route_forwarding"],
    "bindable": true,
    "metadata": {
      "provider": {
        "name": "The name"
      },
      "listing": {
        "imageUrl": "http://example.com/cat.gif",
        "blurb": "Add a blurb here",
        "longDescription": "A long time ago, in a galaxy far far away..."
      },
      "displayName": "The Fake Broker"
    },
    "dashboard_client": {
      "id": "398e2f8e-XXXX-XXXX-XXXX-19a71ecbcf64",
      "secret": "277cabb0-XXXX-XXXX-XXXX-7822c0a90e5d",
      "redirect_uri": "http://localhost:1234"
    },
    "plan_updateable": true,
    "plans": [{
      "name": "fake-plan-1",
      "id": "d3031751-XXXX-XXXX-XXXX-a42377d3320e",
      "description": "Shared fake Server, 5tb persistent disk, 40 max concurrent connections",
      "max_storage_tb": 5,
      "metadata": {
        "costs":[
            {
               "amount":{
                  "usd":99.0
               },
               "unit":"MONTHLY"
            },
            {
               "amount":{
                  "usd":0.99
```

```
                },
                "unit":"1GB of messages over 20GB"
             }
          ],
          "bullets": [
            "Shared fake server",
            "5 TB storage",
            "40 concurrent connections"
          ]
        }
      }, {
        "name": "fake-plan-2",
        "id": "0f4008b5-XXXX-XXXX-XXXX-dace631cd648",
        "description": "Shared fake Server, 5tb persistent disk, 40 max concurrent connections. 100 async",
        "max_storage_tb": 5,
        "metadata": {
          "costs":[
              {
                 "amount":{
                    "usd":199.0
                 },
                 "unit":"MONTHLY"
              },
              {
                 "amount":{
                    "usd":0.99
                 },
                 "unit":"1GB of messages over 20GB"
              }
          ],
          "bullets": [
            "40 concurrent connections"
          ]
        }
      }]
    }]
}
```

Adding a Broker to the Platform

After implementing the first endpoint `GET /v2/catalog` documented above, the service broker will need to be registered with your platform to make your services and plans available to end users.

Synchronous and Asynchronous Operations

Platforms expect prompt responses to all API requests in order to provide users with fast feedback. Service broker authors SHOULD implement their brokers to respond promptly to all requests but will need to decide whether to implement synchronous or asynchronous responses. Brokers that can guarantee completion of the requested operation with the response SHOULD return the synchronous response. Brokers that cannot guarantee completion of the operation with the response SHOULD implement the asynchronous response.

Providing a synchronous response for a provision, update, or bind operation before actual completion causes confusion for users as their service might not be usable and they have no way to find out when it will be. Asynchronous responses set expectations for users that an operation is in progress and can also provide updates on the status of the operation.

Support for synchronous or asynchronous responses MAY vary by service offering, even by service plan.

Synchronous Operations

To execute a request synchronously, the broker need only return the usual status codes: `201 Created` for provision and bind, and `200 OK` for update, unbind, and deprovision.

Brokers that support sychronous responses for provision, update, and delete can ignore the `accepts_incomplete=true` query parameter if it is provided by the client.

Asynchronous Operations

Note: Asynchronous operations are currently supported only for provision, update, and deprovision.

For a broker to return an asynchronous response, the query parameter `accepts_incomplete=true` MUST be included the request. If the parameter is not included or is set to `false`, and the broker cannot fulfill the request synchronously (guaranteeing that the operation is complete on response), then the broker SHOULD reject the request with the status code `422 Unprocessable Entity` and the following body:

```
{
  "error": "AsyncRequired",
  "description": "This service plan requires client support for asynchronous service operations."
}
```

If the query parameter described above is present, and the broker executes the request asynchronously, the broker MUST return the asynchronous response `202 Accepted`. The response body SHOULD be the same as if the broker were serving the request synchronously.

An asynchronous response triggers the platform marketplace to poll the endpoint `GET /v2/service_instances/:guid/last_operation` until the broker indicates that the requested operation has succeeded or failed. Brokers MAY include a status message with each response for the `last_operation` endpoint that provides visibility to end users as to the progress of the operation.

Blocking Operations

Brokers do not have to support concurrent requests that act on the same set of resources. If a broker receives a request that it is not able to process due to other activity being done on that resource then the broker MUST reject the request with an HTTP `422 Unprocessable Entity`. The HTTP body of the response MUST include a `description` property explaining the reason for the failure.

Sample response:

```
{
  "description": "Another operation for this service instance is in progress"
}
```

Polling Last Operation

When a broker returns status code `202 Accepted` for Provision, Update, or Deprovision, the platform will begin polling the `/v2/service_instances/:guid/last_operation` endpoint to obtain the state of the last requested operation. The broker response MUST contain the field `state` and MAY contain the field `description`.

Valid values for `state` are `in progress`, `succeeded`, and `failed`. The platform will poll the `last_operation` endpoint as long as the broker returns `"state": "in progress"`. Returning `"state": "succeeded"` or `"state": "failed"` will cause the platform to cease polling. The value provided for `description` will be passed through to the platform API client and can be used to provide additional detail for users about the progress of the operation.

Request

Route

`GET /v2/service_instances/:instance_id/last_operation`

Parameters

The request provides these query string parameters as useful hints for brokers.

Query-String Field	Type	Description
service_id	string	ID of the service from the catalog. If present, MUST be a non-empty string.
plan_id	string	ID of the plan from the catalog. If present, MUST be a non-empty string.
operation	string	A broker-provided identifier for the operation. When a value for `operation` is included with asynchronous responses for Provision, Update, and Deprovision requests, the platform MUST provide the same value using this query parameter as a URL-encoded string. If present, MUST be a non-empty string.

Note: Although the request query parameters `service_id` and `plan_id` are not mandatory, the platform SHOULD include them on all `last_operation` requests it makes to service brokers.

cURL

```
$ curl http://username:password@broker-url/v2/service_instances/:instance_id/last_operation
```

Response

Status Code	Description
200 OK	MUST be returned upon successful processing of this request. The expected response body is below.
410 Gone	Appropriate only for asynchronous delete operations. The platform MUST consider this response a success and remove the resource from its database. The expected response body is `{}`. Returning this while the platform is polling for create or update operations SHOULD be interpreted as an invalid response and the platform SHOULD continue polling.

Responses with any other status code SHOULD be interpreted as an error or invalid response. The platform SHOULD continue polling until the broker returns a valid response or the maximum polling duration is reached. Brokers MAY use the `description` field to expose user-facing error messages about the operation state; for more info see Broker Errors.

Body

All response bodies MUST be a valid JSON Object (`{}`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body might or might not be returned.

For success responses, the following fields are valid.

Response field	Type	Description
state*	string	Valid values are `in progress`, `succeeded`, and `failed`. While `"state": "in progress"`, the platform SHOULD continue polling. A response with `"state": "succeeded"` or `"state": "failed"` MUST cause the platform to cease polling.
description	string	A user-facing message displayed to the platform API client. Can be used to tell the user details about the status of the operation. If present, MUST be a non-empty string.

* Fields with an asterisk are REQUIRED.

```
{
  "state": "in progress",
```

```
  "description": "Creating service (10% complete)."
}
```

Polling Interval and Duration

The frequency and maximum duration of polling MAY vary by platform client. If a platform has a max polling duration and this limit is reached, the platform MUST cease polling and the operation state MUST be considered `failed`.

Provisioning

When the broker receives a provision request from the platform, it MUST take whatever action is necessary to create a new resource. What provisioning represents varies by service and plan, although there are several common use cases. For a MySQL service, provisioning could result in an empty dedicated database server running on its own VM or an empty schema on a shared database server. For non-data services, provisioning could just mean an account on an multi-tenant SaaS application.

Request

Route

`PUT /v2/service_instances/:instance_id`

The `:instance_id` of a service instance is provided by the platform. This ID will be used for future requests (bind and deprovision), so the broker will use it to correlate the resource it creates.

Parameters

Parameter name	Type	Description
accepts_incomplete	boolean	A value of true indicates that the marketplace and its clients support asynchronous broker operations. If this parameter is not included in the request, and the broker can only provision a service instance of the requested plan asynchronously, the broker MUST reject the request with a `422 Unprocessable Entity` as described below.

Body

Request field	Type	Description
service_id*	string	The ID of the service (from the catalog). MUST be globally unique. MUST be a non-empty string.
plan_id*	string	The ID of the plan (from the catalog) for which the service instance has been requested. MUST be unique to a service. MUST be a non-empty string.
context	object	Platform specific contextual information under which the service instance is to be provisioned. Although most brokers will not use this field, it could be helpful in determining data placement or applying custom business rules. `context` will replace `organization_guid` and `space_guid` in future versions of the specification - in the interim both SHOULD be used to ensure interoperability with old and new implementations.
organization_guid*	string	Deprecated in favor of `context`. The platform GUID for the organization under which the service instance is to be provisioned. Although most brokers will not use this field, it might be helpful for executing operations on a user's behalf. MUST be a non-empty string.
space_guid*	string	Deprecated in favor of `context`. The identifier for the project space within the platform organization. Although most brokers will not use this field, it might be helpful for executing operations on a user's behalf. MUST be a non-empty string.

Request field	Type	Description
parameters	JSON object	Configuration options for the service instance. Controller treats this as a blob. Brokers SHOULD ensure that the client has provided valid configuration parameters and values for the operation.

* Fields with an asterisk are REQUIRED.

```
{
  "context": {
    "platform": "cloudfoundry",
    "some_field": "some-contextual-data"
  },
  "service_id": "service-guid-here",
  "plan_id": "plan-guid-here",
  "organization_guid": "org-guid-here",
  "space_guid": "space-guid-here",
  "parameters": {
    "parameter1": 1,
    "parameter2": "foo"
  }
}
```

cURL

```
$ curl http://username:password@broker-url/v2/service_instances/:instance_id?accepts_incomplete=true -d '{
  "context": {
    "platform": "cloudfoundry",
    "some_field": "some-contextual-data"
  },
  "service_id": "service-guid-here",
  "plan_id": "plan-guid-here",
  "organization_guid": "org-guid-here",
  "space_guid": "space-guid-here",
  "parameters": {
    "parameter1": 1,
    "parameter2": "foo"
  }
}' -X PUT -H "X-Broker-API-Version: 2.12" -H "Content-Type: application/json"
```

Response

Status Code	Description
200 OK	MUST be returned if the service instance already exists, is fully provisioned, and the requested parameters are identical to the existing service instance. The expected response body is below.
201 Created	MUST be returned if the service instance was provisioned as a result of this request. The expected response body is below.
202 Accepted	MUST be returned if the service instance provisioning is in progress. This triggers the platform marketplace to poll the Service Instance Last Operation Endpoint for operation status. Note that a re-sent `PUT` request MUST return a `202 Accepted`, not a `200 OK`, if the service instance is not yet fully provisioned.
409 Conflict	MUST be returned if a service instance with the same id already exists but with different attributes. The expected response body is `{}`, though the description field MAY be used to return a user-facing error message, as described in Broker Errors.

Status Code	Description
422 Unprocessable Entity	MUST be returned if the broker only supports asynchronous provisioning for the requested plan and the request did not include `?accepts_incomplete=true`. The expected response body is: `{ "error": "AsyncRequired", "description": "This service plan requires client support for asynchronous service operations." }`, as described below.

Responses with any other status code will be interpreted as a failure. Brokers can include a user-facing message in the `description` field; for details see Broker Errors.

Body

All response bodies MUST be a valid JSON Object (`{}`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body might or might not be returned.

For success responses, a broker MUST return the following fields. For error responses, see Broker Errors.

Response field	Type	Description
dashboard_url	string	The URL of a web-based management user interface for the service instance; we refer to this as a service dashboard. The URL MUST contain enough information for the dashboard to identify the resource being accessed (`9189kdfsk0vfnku` in the example below). Note: a broker that wishes to return `dashboard_url` for a service instance MUST return it with the initial response to the provision request, even if the service is provisioned asynchronously. If present, MUST be a non-empty string.
operation	string	For asynchronous responses, service brokers MAY return an identifier representing the operation. The value of this field MUST be provided by the platform with requests to the Last Operation endpoint in a URL encoded query parameter. If present, MUST be a non-empty string.

* Fields with an asterisk are REQUIRED.

```
{
  "dashboard_url": "http://example-dashboard.example.com/9189kdfsk0vfnku",
  "operation": "task_10"
}
```

Updating a Service Instance

By implementing this endpoint, service broker authors can enable users to modify two attributes of an existing service instance: the service plan and parameters. By changing the service plan, users can upgrade or downgrade their service instance to other plans. By modifying properties, users can change configuration options that are specific to a service or plan.

To enable support for the update of the plan, a broker MUST declare support per service by including `plan_updateable: true` in its catalog endpoint.

Not all permutations of plan changes are expected to be supported. For example, a service might support upgrading from plan "shared small" to "shared large" but not to plan "dedicated". It is up to the broker to validate whether a particular permutation of plan change is supported. If a particular plan change is not supported, the broker SHOULD return a meaningful error message in response.

Request

Route

`PATCH /v2/service_instances/:instance_id`

`:instance_id` is the global unique ID of a previously provisioned service instance.

Parameters

Parameter name	Type	Description
accepts_incomplete	boolean	A value of true indicates that the marketplace and its clients support asynchronous broker operations. If this parameter is not included in the request, and the broker can only provision a service instance of the requested plan asynchronously, the broker SHOULD reject the request with a `422 Unprocessable Entity` as described below.

Body

Request Field	Type	Description
context	object	Contextual data under which the service instance is created.
service_id*	string	The ID of the service (from the catalog). MUST be globally unique. MUST be a non-empty string.
plan_id	string	The ID of the plan (from the catalog) for which the service instance has been requested. MUST be unique to a service. If present, MUST be a non-empty string. If this field is not present in the request message, then the broker MUST NOT change the plan of the instance as a result of this request.
parameters	JSON object	Configuration options for the service instance. An opaque object, controller treats this as a blob. Brokers SHOULD ensure that the client has provided valid configuration parameters and values for the operation. If this field is not present in the request message, then the broker MUST NOT change the parameters of the instance as a result of this request.
previous_values	object	Information about the service instance prior to the update.
previous_values.service_id	string	Deprecated; determined to be unnecessary as the value is immutable. ID of the service for the service instance. If present, MUST be a non-empty string.
previous_values.plan_id	string	ID of the plan prior to the update. If present, MUST be a non-empty string.
previous_values.organization_id	string	Deprecated as it was redundant information. Organization for the service instance MUST be provided by platforms in the top-level field `context`. ID of the organization specified for the service instance. If present, MUST be a non-empty string.
previous_values.space_id	string	Deprecated as it was redundant information. Space for the service instance MUST be provided by platforms in the top-level field `context`. ID of the space specified for the service instance. If present, MUST be a non-empty string.

* Fields with an asterisk are REQUIRED.

```
{
  "context": {
    "platform": "cloudfoundry",
    "some_field": "some-contextual-data"
  },
  "service_id": "service-guid-here",
  "plan_id": "plan-guid-here",
  "parameters": {
    "parameter1": 1,
    "parameter2": "foo"
  },
  "previous_values": {
    "plan_id": "old-plan-guid-here",
```

```
    "service_id": "service-guid-here",
    "organization_id": "org-guid-here",
    "space_id": "space-guid-here"
  }
}
```

cURL

```
$ curl http://username:password@broker-url/v2/service_instances/:instance_id?accepts_incomplete=true -d '{
  "context": {
    "platform": "cloudfoundry",
    "some_field": "some-contextual-data"
  },
  "service_id": "service-guid-here",
  "plan_id": "plan-guid-here",
  "parameters": {
    "parameter1": 1,
    "parameter2": "foo"
  },
  "previous_values": {
    "plan_id": "old-plan-guid-here",
    "service_id": "service-guid-here",
    "organization_id": "org-guid-here",
    "space_id": "space-guid-here"
  }
}' -X PATCH -H "X-Broker-API-Version: 2.12" -H "Content-Type: application/json"
```

Response

Status Code	Description
200 OK	MUST be returned if the request's changes have been applied. The expected response body is `{}`.
202 Accepted	MUST be returned if the service instance update is in progress. This triggers the platform marketplace to poll the Last Operation for operation status. Note that a re-sent `PATCH` request MUST return a `202 Accepted`, not a `200 OK`, if the requested update has not yet completed.
422 Unprocessable entity	MUST be returned if the requested change is not supported or if the request cannot currently be fulfilled due to the state of the service instance (e.g. service instance utilization is over the quota of the requested plan). Brokers SHOULD include a user-facing message in the body; for details see Broker Errors. Additionally, a `422 Unprocessable Entity` can also be returned if the broker only supports asynchronous update for the requested plan and the request did not include `?accepts_incomplete=true`; in this case the expected response body is: `{ "error": "AsyncRequired", "description": "This service plan requires client support for asynchronous service operations." }`

Responses with any other status code will be interpreted as a failure. Brokers can include a user-facing message in the `description` field; for details see Broker Errors.

Body

All response bodies MUST be a valid JSON Object (`{}`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body might or might not be returned.

For success responses, a broker MUST return the following field. Others will be ignored. For error responses, see Broker Errors.

Response field	Type	Description

Response field	Type	Description
operation	string	For asynchronous responses, service brokers MAY return an identifier representing the operation. The value of this field MUST be provided by the platform with requests to the Last Operation endpoint in a URL encoded query parameter. If present, MUST be a non-empty string.

* Fields with an asterisk are REQUIRED.

```
{
  "operation": "task_10"
}
```

Binding

If `bindable:true` is declared for a service or plan in the Catalog endpoint, the platform MAY request generation of a service binding.

Note: Not all services need to be bindable --- some deliver value just from being provisioned. Brokers that offer services that are bindable MUST declare them as such using `bindable: true` in the [Catalog](#catalog-management). Brokers that do not offer any bindable services do not need to implement the endpoint for bind requests.

Types of Binding

Credentials

Credentials are a set of information used by an application or a user to utilize the service instance. If the broker supports generation of credentials it MUST return `credentials` in the response for a request to create a service binding. Credentials SHOULD be unique whenever possible, so access can be revoked for each binding without affecting consumers of other bindings for the service instance.

Log Drain

There are a class of service offerings that provide aggregation, indexing, and analysis of log data. To utilize these services an application that generates logs needs information for the location to which it will stream logs. A create binding response from a service broker that provides one of these services MUST include a `syslog_drain_url`. The platform MUST use the `syslog_drain_url` value when sending logs to the service.

Brokers MUST NOT include a `syslog_drain_url` in a create binding response if the associated Catalog entry for the service did not include a `"requires":["syslog_drain"]` property.

Route Services

There are a class of service offerings that intermediate requests to applications, performing functions such as rate limiting or authorization.

If a platform supports route services, it MUST send a routable address, or endpoint, for the application along with the request to create a service binding using `"bind_resource":{"route":"some-address.com"}`. A broker MAY support configuration specific to an address using parameters; exposing this feature to users would require a platform to support binding multiple routable addresses to the same service instance.

If a service is deployed in a configuration to support this behavior, the broker MUST return a `route_service_url` in the response for a request to create a binding, so that the platform knows where to proxy the application request. If the service is deployed such that the network configuration to proxy application requests through instances of the service is managed out-of-band, the broker MUST NOT return `route_service_url` in the response.

Brokers MUST NOT include a `route_service_url` in a create binding response if the associated Catalog entry for the service did not include a `"requires":["route_forwarding"]` property.

Volume Services

There are a class of services that provide network storage to applications via volume mounts in the application container. A create binding response from one of these services MUST include `volume_mounts`.

Brokers MUST NOT include `volume_mounts` in a create binding response if the associated Catalog entry for the service did not include a `"requires":["volume_mount"]` property.

Request

Route

`PUT /v2/service_instances/:instance_id/service_bindings/:binding_id`

The `:instance_id` is the ID of a previously provisioned service instance. The `:binding_id` is also provided by the platform. This ID will be used for future unbind requests, so the broker will use it to correlate the resource it creates.

Body

Request Field	Type	Description
service_id*	string	ID of the service from the catalog. MUST be a non-empty string.
plan_id*	string	ID of the plan from the catalog. MUST be a non-empty string.
app_guid	string	Deprecated in favor of `bind_resource.app_guid`. GUID of an application associated with the binding to be created. If present, MUST be a non-empty string.
bind_resource	JSON object	A JSON object that contains data for platform resources associated with the binding to be created. See Bind Resource Object for more information.
parameters	JSON object	Configuration options for the service binding. An opaque object, controller treats this as a blob. Brokers SHOULD ensure that the client has provided valid configuration parameters and values for the operation.

Bind Resource Object

The `bind_resource` object contains platform specific information related to the context in which the service will be used. In some cases the platform might not be able to provide this information at the time of the binding request, therefore the `bind_resource` and its fields are OPTIONAL.

Below are some common fields that MAY be used. Platforms MAY choose to add additional ones as needed.

Request Field	Type	Description
app_guid	string	GUID of an application associated with the binding. For credentials bindings.
route	string	URL of the application to be intermediated. For route services bindings.

* Fields with an asterisk are REQUIRED.

```
{
  "service_id": "service-guid-here",
  "plan_id": "plan-guid-here",
  "bind_resource": {
    "app_guid": "app-guid-here"
  },
  "parameters": {
    "parameter1-name-here": 1,
    "parameter2-name-here": "parameter2-value-here"
  }
}
```

cURL

```
$ curl http://username:password@broker-url/v2/service_instances/:instance_id/service_bindings/:binding_id -d '{
  "service_id": "service-guid-here",
  "plan_id": "plan-guid-here",
  "bind_resource": {
    "app_guid": "app-guid-here"
  },
  "parameters": {
    "parameter1-name-here": 1,
    "parameter2-name-here": "parameter2-value-here"
  }
}' -X PUT
```

Response

Status Code	Description
200 OK	MUST be returned if the binding already exists and the requested parameters are identical to the existing binding. The expected response body is below.
201 Created	MUST be returned if the binding was created as a result of this request. The expected response body is below.
409 Conflict	MUST be returned if a service binding with the same id, for the same service instance, already exists but with different parameters. The expected response body is `{}`, though the description field MAY be used to return a user-facing error message, as described in Broker Errors.
422 Unprocessable Entity	MUST be returned if the broker requires that `app_guid` be included in the request body. The expected response body is: `{ "error": "RequiresApp", "description": "This service supports generation of credentials through binding an application only." }`

Responses with any other status code will be interpreted as a failure and an unbind request will be sent to the broker to prevent an orphan being created on the broker. Brokers can include a user-facing message in the `description` field; for details see Broker Errors.

Body

All response bodies MUST be a valid JSON Object (`{}`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body might or might not be returned.

For success responses, the following fields are supported. Others will be ignored. For error responses, see Broker Errors.

Response Field	Type	Description
credentials	object	A free-form hash of credentials that can be used by applications or users to access the service.
syslog_drain_url	string	A URL to which logs MUST be streamed. `"requires":["syslog_drain"]` MUST be declared in the Catalog endpoint or the platform MUST consider the response invalid.
route_service_url	string	A URL to which the platform MUST proxy requests for the address sent with `bind_resource.route` in the request body. `"requires":["route_forwarding"]` MUST be declared in the Catalog endpoint or the platform can consider the response invalid.
volume_mounts	array-of-objects	An array of configuration for remote storage devices to be mounted into an application container filesystem. `"requires":["volume_mount"]` MUST be declared in the Catalog endpoint or the platform can consider the response invalid.

Volume Mounts Object

Response Field	Type	Description
driver*	string	Name of the volume driver plugin which manages the device.
container_dir*	string	The path in the application container onto which the volume will be mounted. This specification does not mandate what action the platform is to take if the path specified already exists in the container.
mode*	string	"r" to mount the volume read-only or "rw" to mount it read-write.
device_type*	string	A string specifying the type of device to mount. Currently the only supported value is "shared".
device*	device-object	Device object containing device_type specific details. Currently only shared devices are supported.

Device Object

Currently only shared devices are supported; a distributed file system which can be mounted on all app instances simultaneously.

Field	Type	Description
volume_id*	string	ID of the shared volume to mount on every app instance.
mount_config	object	Configuration object to be passed to the driver when the volume is mounted.

* Fields with an asterisk are REQUIRED.

```
{
  "credentials": {
    "uri": "mysql://mysqluser:pass@mysqlhost:3306/dbname",
    "username": "mysqluser",
    "password": "pass",
    "host": "mysqlhost",
    "port": 3306,
    "database": "dbname"
  }
}
```

```
{
  "volume_mounts": [{
    "driver": "cephdriver",
    "container_dir": "/data/images",
    "mode": "r",
    "device_type": "shared",
    "device": {
      "volume_id": "bc2c1eab-05b9-482d-b0cf-750ee07de311",
      "mount_config": {
        "key": "value"
      }
    }
  }]
}
```

Unbinding

Note: Brokers that do not provide any bindable services or plans do not need to implement this endpoint.

When a broker receives an unbind request from the marketplace, it MUST delete any resources associated with the binding. In the case where credentials were generated, this might result in requests to the service instance failing to authenticate.

Request

Route

`DELETE /v2/service_instances/:instance_id/service_bindings/:binding_id`

The `:instance_id` is the ID of a previously provisioned service instance. The `:binding_id` is the ID of a previously provisioned binding for that service instance.

Parameters

The request provides these query string parameters as useful hints for brokers.

Query-String Field	Type	Description
service_id*	string	ID of the service from the catalog. MUST be a non-empty string.
plan_id*	string	ID of the plan from the catalog. MUST be a non-empty string.

* Query parameters with an asterisk are REQUIRED.

cURL

```
$ curl 'http://username:password@broker-url/v2/service_instances/:instance_id/
  service_bindings/:binding_id?service_id=service-id-here&plan_id=plan-id-here' -X DELETE -H "X-Broker-API-
Version: 2.12"
```

Response

Status Code	Description
200 OK	MUST be returned if the binding was deleted as a result of this request. The expected response body is `{}`.
410 Gone	MUST be returned if the binding does not exist. The expected response body is `{}`.

Responses with any other status code will be interpreted as a failure and the binding will remain in the marketplace database. Brokers can include a user-facing message in the `description` field; for details see Broker Errors.

Body

All response bodies MUST be a valid JSON Object (`{}`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body might or might not be returned.

For a success response, the expected response body is `{}`.

Deprovisioning

When a broker receives a deprovision request from the marketplace, it MUST delete any resources it created during the provision. Usually this means that all resources are immediately reclaimed for future provisions.

Request

Route

`DELETE /v2/service_instances/:instance_id`

`:instance_id` is the identifier of a previously provisioned service instance.

Parameters

The request provides these query string parameters as useful hints for brokers.

Query-String Field	Type	Description
service_id*	string	ID of the service from the catalog. MUST be a non-empty string.
plan_id*	string	ID of the plan from the catalog. MUST be a non-empty string.
accepts_incomplete	boolean	A value of true indicates that both the marketplace and the requesting client support asynchronous deprovisioning. If this parameter is not included in the request, and the broker can only deprovision a service instance of the requested plan asynchronously, the broker MUST reject the request with a `422 Unprocessable Entity` as described below.

* Query parameters with an asterisk are REQUIRED.

cURL

```
$ curl 'http://username:password@broker-url/v2/service_instances/:instance_id?accepts_incomplete=true
  &service_id=service-id-here&plan_id=plan-id-here' -X DELETE -H "X-Broker-API-Version: 2.12"
```

Response

Status Code	Description
200 OK	MUST be returned if the service instance was deleted as a result of this request. The expected response body is `{}`.
202 Accepted	MUST be returned if the service instance deletion is in progress. This triggers the marketplace to poll the Service Instance Last Operation Endpoint for operation status. Note that a re-sent `DELETE` request MUST return a `202 Accepted`, not a `200 OK`, if the delete request has not completed yet.
410 Gone	MUST be returned if the service instance does not exist. The expected response body is `{}`.
422 Unprocessable Entity	MUST be returned if the broker only supports asynchronous deprovisioning for the requested plan and the request did not include `?accepts_incomplete=true`. The expected response body is: `{ "error": "AsyncRequired", "description": "This service plan requires client support for asynchronous service operations." }`, as described below.

Responses with any other status code will be interpreted as a failure and the service instance will remain in the marketplace database. Brokers can include a user-facing message in the `description` field; for details see Broker Errors.

Body

All response bodies MUST be a valid JSON Object (`{}`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body might or might not be returned.

For success responses, the following fields are supported. Others will be ignored. For error responses, see Broker Errors.

Response field	Type	Description
operation	string	For asynchronous responses, service brokers MAY return an identifier representing the operation. The value of this field MUST be provided by the platform with requests to the Last Operation endpoint in a URL encoded query parameter. If present, MUST be a non-empty string.

* Fields with an asterisk are REQUIRED.

```
{
  "operation": "task_10"
```

```
}
```

Broker Errors

Response

Broker failures beyond the scope of the well-defined HTTP response codes listed above (like `410 Gone` on Deprovisioning) MUST return an appropriate HTTP response code (chosen to accurately reflect the nature of the failure) and a body containing a valid JSON Object (not an array).

Body

All response bodies MUST be a valid JSON Object (`{}`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body might or might not be returned.

For error responses, the following fields are valid. Others will be ignored. If an empty JSON object is returned in the body `{}`, a generic message containing the HTTP response code returned by the broker will be displayed to the requester.

Response Field	Type	Description
description	string	A meaningful error message explaining why the request failed.

```
{
  "description": "Your account has exceeded its quota for service instances. Please contact support at
http://support.example.com."
}
```

Orphans

The platform marketplace is the source of truth for service instances and bindings. Service brokers are expected to have successfully provisioned all the service instances and bindings that the marketplace knows about, and none that it doesn't.

Orphans can result if the broker does not return a response before a request from the marketplace times out (typically 60 seconds). For example, if a broker does not return a response to a provision request before the request times out, the broker might eventually succeed in provisioning a service instance after the marketplace considers the request a failure. This results in an orphan service instance on the broker's side.

To mitigate orphan service instances and bindings, the marketplace SHOULD attempt to delete resources it cannot be sure were successfully created, and SHOULD keep trying to delete them until the broker responds with a success.

Platforms SHOULD initiate orphan mitigation in the following scenarios:

Status code of broker response	Platform interpretation of response	Platform initiates orphan mitigation?
200	Success	No
200 with malformed response	Failure	No
201	Success	No
201 with malformed response	Failure	Yes
All other 2xx	Failure	Yes
408	Failure due to timeout	Yes
All other 4xx	Broker rejected request	No
5xx	Broker error	Yes

Status code of broker response	Platform interpretation of response	Platform initiates orphan mitigation?
Timeout	Failure	Yes

If the platform marketplace encounters an internal error provisioning a service instance or binding (for example, saving to the database fails), then it MUST at least send a single delete or unbind request to the service broker to prevent creation of an orphan.