The API request lifecycle

Flexible and Annual Redis Enterprise Cloud subscriptions can leverage a RESTful API that permits operations against a variety of resources, including servers, services, and related infrastructure.

```
Note: The REST API is not supported for Fixed or Free subscriptions.
```

Once it’s enabled, you can use the REST API to create, update, and delete subscriptions, databases, and other entities.

API operations run asynchronously, which means that provisioning occurs in the background. When you submit a request, a background process starts working on it. The response object includes an ID that lets you determine the status of the background process as it performs its work.

For operations that do not create or modify resources (such as most GET operations), the API is synchronous; that is, the response object reports the results of the request.

Asynchronous operations have two main phases: processing and provisioning. A resource is not available until both phases are complete.

```
Task processing

During this phase, the request is received, evaluated, planned, and executed.

Use tasks to track requests

Many operations are asychronous, including CREATE, UPDATE, and DELETE operations. The response objects for such operations provide a taskId identifier that lets you track the progress of the underlying operation.

You can query the taskId to track the state of a specific task:

```
curl -s -X GET "https://$HOST/tasks/$TASK_ID" \
   -H "accept: application/json" \
   -H "x-api-key: $ACCOUNT_KEY" \
   -H "x-api-secret-key: $SECRET_KEY" \
   | jq -r .
```
```
Note: The REST API is not supported for Fixed or Free subscriptions.
```
In this example, the $TASK_ID variable includes the taskId value.

```
TASK_ID=166d7f69-f35b-41ed-9128-7d753b642d63
```

You can also query the state of all active tasks or recently completed tasks in your account:

```
curl -s -X GET "https://$HOST/tasks" \
  -H "accept: application/json" \
  -H "x-api-key: $ACCOUNT_KEY" \
  -H "x-api-secret-key: $SECRET_KEY" \
  | jq -r .
```

### Task process states

During the processing of a request, the task moves through these states:

- **received** - Request is received and awaits processing.
- **processing-in-progress** - A dedicated worker is processing the request.
- **processing-completed** - Request processing succeeded and the request is being provisioned (or de-provisioned, depending on the specific request). A response segment is included with the task status JSON response. The response includes a resourceId for each resource that the request creates, such as Subscription or Database ID.
- **processing-error** - Request processing failed. A detailed cause or reason is included in the task status JSON response.

**Note:** A task that reaches the received state cannot be cancelled and it will await completion (i.e., processing and provisioning). If you wish to undo an operation that was performed by a task, perform a compensating action (for example: delete a subscription that was created unintentionally).

### Task provisioning phase

When the processing phase succeeds and the task is in the processing-completed state, the provisioning phase starts. During the provisioning phase, the API orchestrates all of the infrastructure, resources, and dependencies required by the request.

**Note:** The term "provisioning" refers to all infrastructure changes required in order to apply the request. This includes provisioning new or additional infrastructure.

The provisioning phase may require several minutes to complete. You can query the resource identifier to track the progress of the provisioning phase.

For example, when you provision a new subscription, use this cURL command to query the status of the subscription:

```
curl -s -X GET "https://$HOST/subscriptions/{subscription-id}" \
  -H "accept: application/json" \
  -H "x-api-key: $ACCOUNT_KEY" \
  -H "x-api-secret-key: $SECRET_KEY" \
  | jq -r .
```

Where the `{subscription-id}` is the resource ID that you receive when the task is in the processing-completed state.

### Provisioning state values
During the provisioning of a resource (such as a subscription, database, or cloud account) the resource transitions through these states:

- **pending** - Provisioning is in progress.
- **active** - Provisioning completed successfully.
- **deleting** - De-provisioning and deletion is in progress.
- **error** - An error occurred during the provisioning phase, including the details of the error.

**Process limitations**

The following limitations apply to asynchronous operations:

- For each account, only one operation is **processed** concurrently. When multiple tasks are sent for the same account, they will be received and processed one after the other.

- The provisioning phase can be performed in parallel except:
  - Subscription creation, update, and deletion: You cannot change (make non-active) more than three subscriptions at the same time.
  - Database creation in an existing subscription: This can cause the subscription state to change from **active** to **pending** during database provisioning in cases such as database sizing that requires cluster resizing or updating cluster metadata.

- For example:
  - Concurrently sending multiple "create database" tasks will cause each task to be in the **received** state, awaiting processing.
  - When the first task starts processing it will be moved to the **processing-in-progress** state.
  - When that first task is completed (either **processing-completed** or **processing-error**), the second task will start processing, and so on.
  - Typically, the processing phase is much faster than the provisioning phase, and multiple tasks will be in provisioned concurrently.
  - If the creation of the database requires an update to the subscription, the subscription state is set to **pending**. When you create multiple databases one after the other, we recommend that you check the subscription state after the processing phase of each database create request. If the subscription is in **pending** state you must wait for the subscription changes to complete and the subscription state to return to **active**.

**Updated:** July 9, 2021