**Backup and recover language understanding models**

When you create a Language resource in the Azure portal, you specify a region for it to be created in. From then on, your resource and all of the operations related to it take place in the specified Azure server region. It's rare, but not impossible, to encounter a network issue that hits an entire region. If your solution needs to always be available, then you should design it to either fail-over into another region. This requires two Azure AI Language resources in different regions and the ability to sync your CLU models across regions.

If your app or business depends on the use of a CLU model, we recommend that you create a replica of your project into another supported region. So that if a regional outage occurs, you can then access your model in the other fail-over region where you replicated your project.

Replicating a project means that you export your project metadata and assets and import them into a new project. This only makes a copy of your project settings, intents, entities and utterances. You still need to [train](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/how-to/train-model) and [deploy](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/how-to/deploy-model) the models to be available for use with [runtime APIs](https://aka.ms/clu-apis).

**Prerequisites**

* Two Azure AI Language resources in different Azure regions, each of them in a different region.

**Export your primary project assets**

Start by exporting the project assets from the project in your primary resource.

**Submit export job**

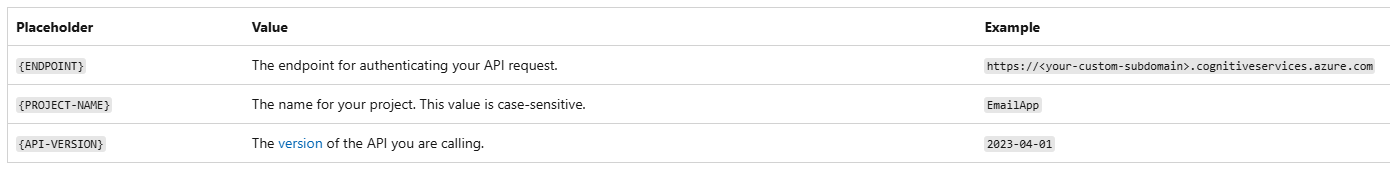
Replace the placeholders in the following request with your {PRIMARY-ENDPOINT} and {PRIMARY-RESOURCE-KEY} that you obtained in the first step.

Create a **POST** request using the following URL, headers, and JSON body to export your project.

**Request URL**

Use the following URL when creating your API request. Replace the placeholder values below with your own values.



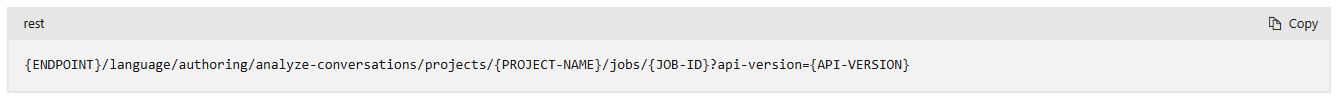


**Headers**

Use the following header to authenticate your request.



Once you send your API request, you will receive a 202 response indicating success. In the response headers, extract the operation-location value. It will be formatted like this:



**JOB-ID** is used to identify your request, since this operation is asynchronous. Use this URL to get the exported project JSON, using the same authentication method.

**Get export job status**

Replace the placeholders in the following request with your {PRIMARY-ENDPOINT} and {PRIMARY-RESOURCE-KEY} that you obtained in the first step.

Use the following **GET** request to query the status of your export job. You can use the URL you received from the previous step, or replace the placeholder values below with your own values.

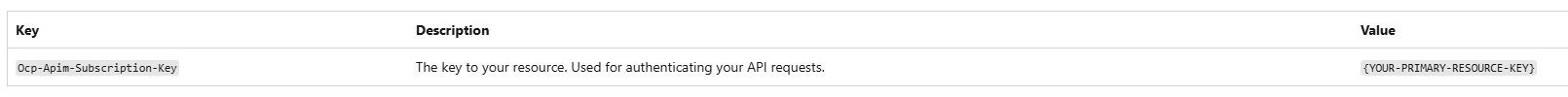


**A screenshot of a computer

Description automatically generated**

**Headers**

Use the following header to authenticate your request.



**Response body**

A close-up of a computer screen

Description automatically generated

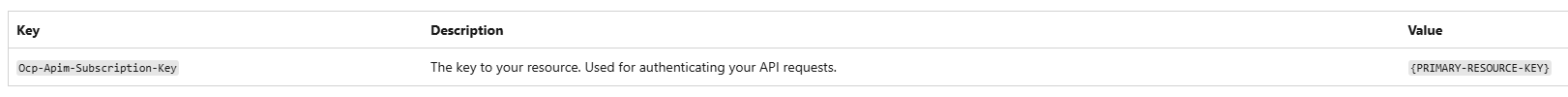
Use the url from the **resultUrl** key in the body to view the exported assets from this job.

**Get export results**

Submit a **GET** request using the {RESULT-URL} you received from the previous step to view the results of the export job.

**Headers**

Use the following header to authenticate your request.



Copy the response body as you will use it as the body for the next import job.

**Import to a new project**

Now go ahead and import the exported project assets in your new project in the secondary region so you can replicate it.

**Submit import job**

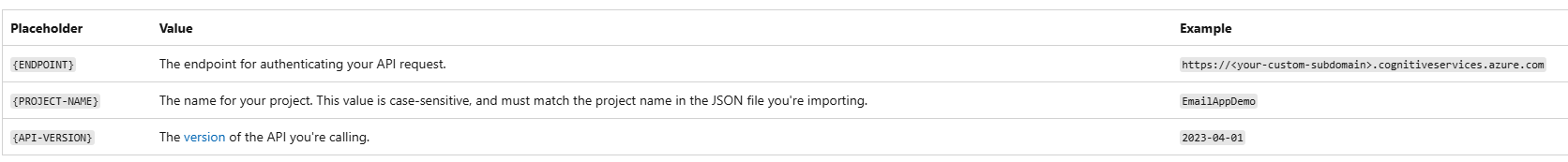
Replace the placeholders in the following request with your {SECONDARY-ENDPOINT} and {SECONDARY-RESOURCE-KEY} that you obtained in the first step.

Submit a **POST** request using the following URL, headers, and JSON body to import your project.

**Request URL**

Use the following URL when creating your API request. Replace the placeholder values with your own values.





**Headers**

Use the following header to authenticate your request.



**Body**

The JSON body you send is similar to the following example.

A screenshot of a computer

Description automatically generated

A white background with black and red lines

Description automatically generated with medium confidence

A close-up of a text

Description automatically generated

Upon a successful request, the API response will contain an operation-location header with a URL you can use to check the status of the import job. It is formatted like this:



**Get import job status**

Replace the placeholders in the following request with your {SECONDARY-ENDPOINT} and {SECONDARY-RESOURCE-KEY} that you obtained in the first step.

When you send a successful project import request, the full request URL for checking the import job's status (including your endpoint, project name, and job ID) is contained in the response's operation-location header.

Use the following **GET** request to query the status of your import job. You can use the URL you received from the previous step, or replace the placeholder values with your own values.

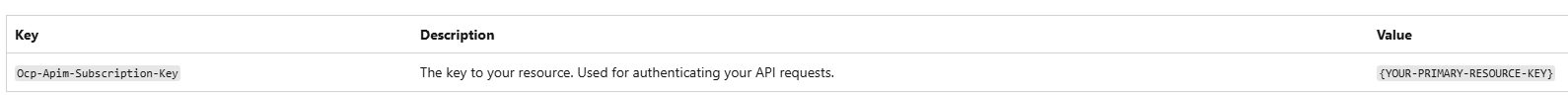


A white rectangular object with a black border

Description automatically generated with medium confidence

**Headers**

Use the following header to authenticate your request.



**Response body**

Once you send the request, you'll get the following response. Keep polling this endpoint until the status parameter changes to "succeeded".

A close-up of a white box

Description automatically generated

**Train your model**

After importing your project, you only have copied the project's assets and metadata and assets. You still need to train your model, which will incur usage on your account.

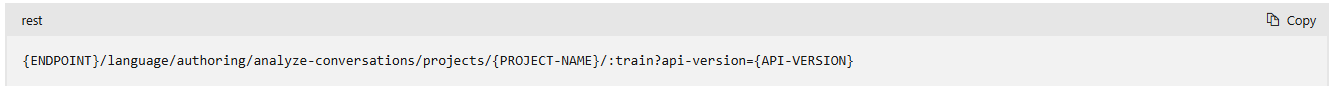
**Submit training job**

Replace the placeholders in the following request with your {SECONDARY-ENDPOINT} and {SECONDARY-RESOURCE-KEY} that you obtained in the first step.

Create a **POST** request using the following URL, headers, and JSON body to submit a training job.

**Request URL**

Use the following URL when creating your API request. Replace the placeholder values with your own values.



A white background with black text

Description automatically generated

**Headers**

Use the following header to authenticate your request.



**Request body**

Use the following object in your request. The model will be named after the value you use for the modelLabel parameter once training is complete.

A white rectangular object with a black border

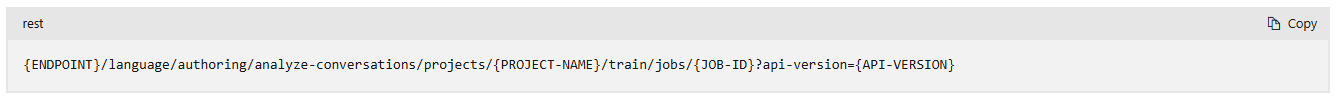
Description automatically generated

A screenshot of a computer

Description automatically generated

**Note :** The trainingSplitPercentage and testingSplitPercentage are only required if Kind is set to percentage and the sum of both percentages should be equal to 100.

Once you send your API request, you will receive a 202 response indicating success. In the response headers, extract the operation-location value. It will be formatted like this:



You can use this URL to get the training job status.

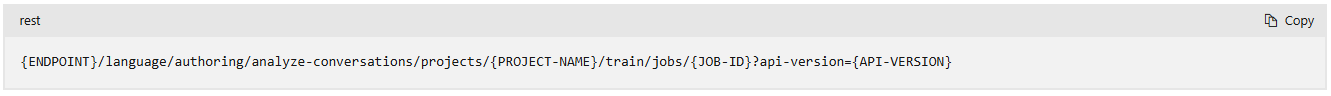
**Get Training Status**

Replace the placeholders in the following request with your {SECONDARY-ENDPOINT} and {SECONDARY-RESOURCE-KEY} that you obtained in the first step.

When you send a successful training request, the full request URL for checking the job's status (including your endpoint, project name, and job ID) is contained in the response's operation-location header.

Use the following **GET** request to get the status of your model's training progress. Replace the placeholder values below with your own values.

**Request URL**



A screenshot of a computer

Description automatically generated

**Headers**

Use the following header to authenticate your request.



**Response Body**

Once you send the request, you will get the following response. Keep polling this endpoint until the **status** parameter changes to "succeeded".

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Deploy your model**

This is the step where you make your trained model available form consumption via the [runtime prediction API](https://aka.ms/ct-runtime-swagger).

**Submit deployment job**

Replace the placeholders in the following request with your {SECONDARY-ENDPOINT} and {SECONDARY-RESOURCE-KEY} that you obtained in the first step.

Create a **PUT** request using the following URL, headers, and JSON body to start deploying a conversational language understanding model.

**Request URL**



A screenshot of a computer

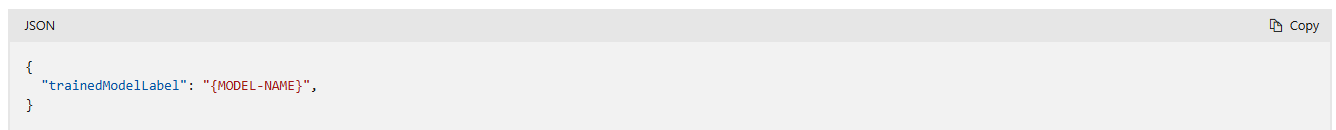
Description automatically generated

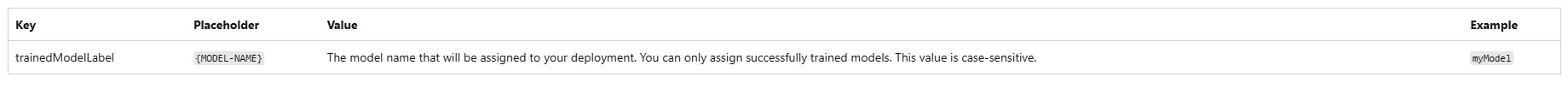
**Headers**

Use the following header to authenticate your request.

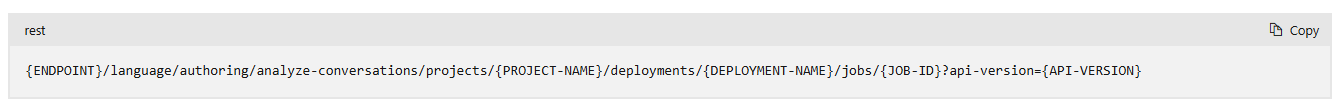


**Request Body**





Once you send your API request, you will receive a 202 response indicating success. In the response headers, extract the operation-location value. It will be formatted like this:



You can use this URL to get the deployment job status.

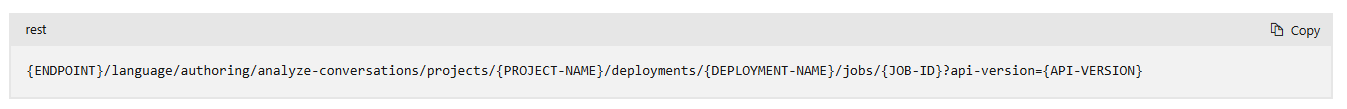
**Get the deployment status**

Replace the placeholders in the following request with your {SECONDARY-ENDPOINT} and {SECONDARY-RESOURCE-KEY} that you obtained in the first step.

When you send a successful deployment request, the full request URL for checking the job's status (including your endpoint, project name, and job ID) is contained in the response's operation-location header.

Use the following **GET** request to get the status of your deployment job. Replace the placeholder values with your own values.

**Request URL**



A screenshot of a computer

Description automatically generated

**Headers**

Use the following header to authenticate your request.



**Response Body**

Once you send the request, you'll get the following response. Keep polling this endpoint until the **status** parameter changes to "succeeded".

A white rectangular object with a black border

Description automatically generated

**Changes in calling the runtime**

Within your system, at the step where you call [runtime API](https://aka.ms/clu-apis) check for the response code returned from the submit task API. If you observe a **consistent** failure in submitting the request, this could indicate an outage in your primary region. Failure once doesn't mean an outage, it may be transient issue. Retry submitting the job through the secondary resource you have created. For the second request use your {YOUR-SECONDARY-ENDPOINT} and secondary key, if you have followed the steps above, {PROJECT-NAME} and {DEPLOYMENT-NAME} would be the same so no changes are required to the request body.

In case you revert to using your secondary resource you will observe slight increase in latency because of the difference in regions where your model is deployed.

**Check if your projects are out of sync**

Maintaining the freshness of both projects is an important part of process. You need to frequently check if any updates were made to your primary project so that you move them over to your secondary project. This way if your primary region fail and you move into the secondary region you should expect similar model performance since it already contains the latest updates. Setting the frequency of checking if your projects are in sync is an important choice, we recommend that you do this check daily in order to guarantee the freshness of data in your secondary model.

**Get project details**

Use the following url to get your project details, one of the keys returned in the body indicates the last modified date of the project. Repeat the following step twice, one for your primary project and another for your secondary project and compare the timestamp returned for both of them to check if they are out of sync.

Use the following **GET** request to get your project details. You can use the URL you received from the previous step, or replace the placeholder values below with your own values.

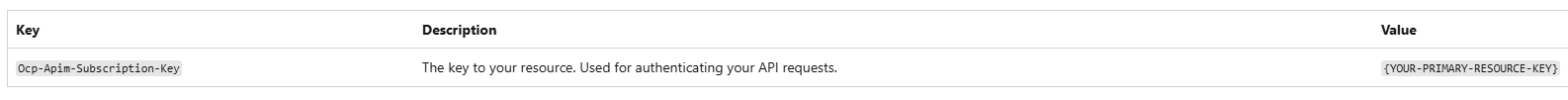


A white line on a white surface

Description automatically generated

**Headers**

Use the following header to authenticate your request.



**Response body**

A white rectangular object with a black border

Description automatically generated

Repeat the same steps for your replicated project using {SECONDARY-ENDPOINT} and {SECONDARY-RESOURCE-KEY}. Compare the returned lastModifiedDateTime from both projects. If your primary project was modified sooner than your secondary one, you need to repeat the steps of [exporting](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/how-to/fail-over#export-your-primary-project-assets), [importing](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/how-to/fail-over#import-to-a-new-project), [training](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/how-to/fail-over#train-your-model) and [deploying](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/how-to/fail-over#deploy-your-model) your model.