Jobs, Reviews, and Workflows

Eduardo Freitas

DATA CAPTURE SPECIALIST

Overview

Job Operations API Methods

Review API Methods

Workflow API Methods

Review Tool Demo

Code Demo

Summary

What Are Job Operations?

They initiate end-to-end moderation jobs with a human-in-loop

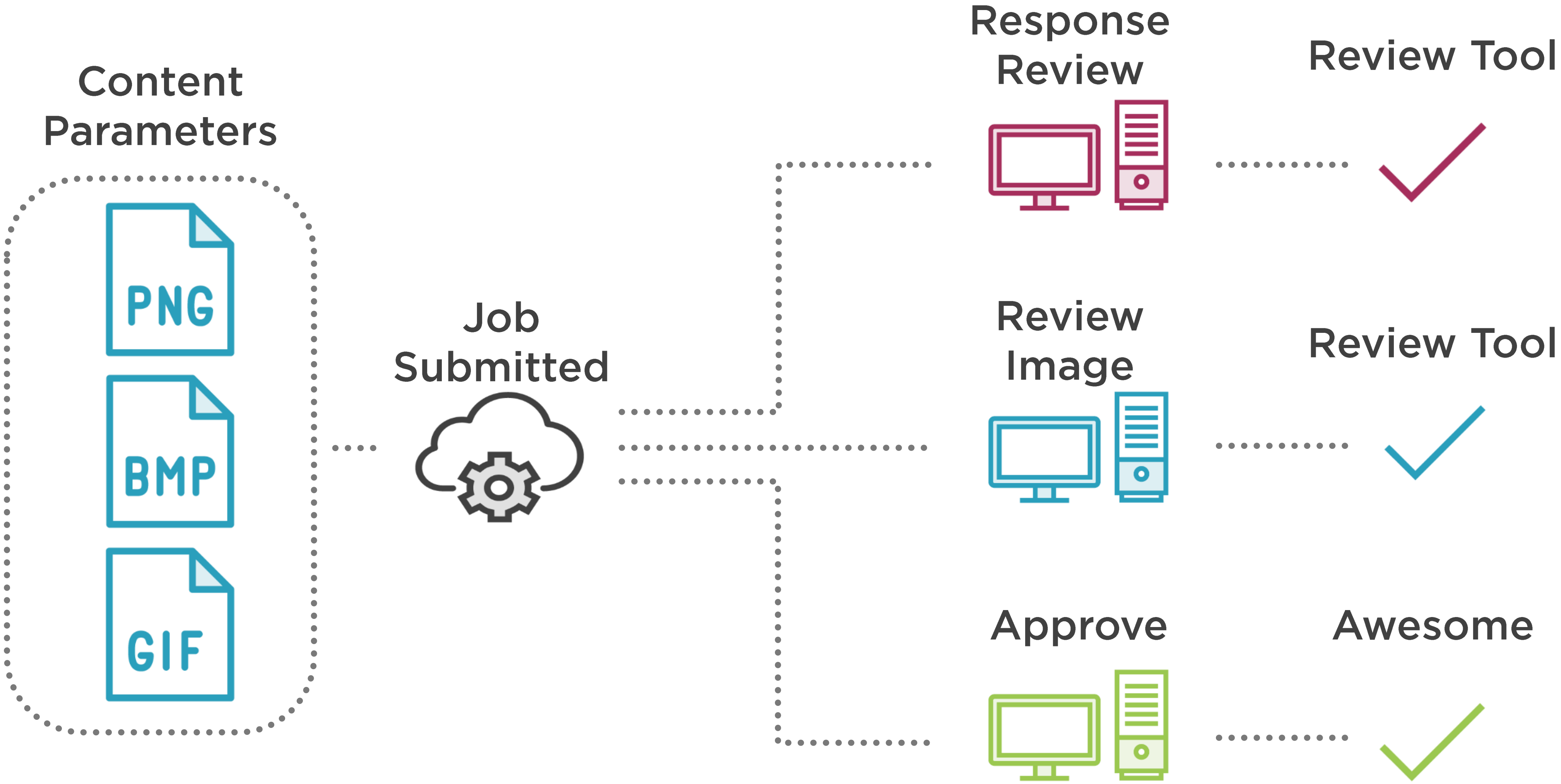
Job Operations Prerequisites

Review Tool Sign-up

Define a Custom Workflow

Create a Job Operation

Content Moderation Job Process



What Are Review Operations?

Human moderators can use the Review Tool to review submitted content

Review Operations Prerequisites

Review Tool Sign-up

Create a Review Operation

Job API Methods: Create

```
curl -v -X POST
"https://[location].api.cognitive.microsoft.com/contentmode
rator/review/v1.0/teams/{teamName}/jobs[?ContentType][&Cont
entId][&WorkflowName][&CallBackEndpoint]"

-H "Content-Type: application/json"

-H "Ocp-Apim-Subscription-Key: {subscription key}"

--data-ascii "{body}"
```

Job API - Create Method

- location
- Ocp-Apim-Subscription-Key
- {body}

Content-Type
image/jpeg
application/json

Using Raw C#

```
public static async void MakeRequest()
{
     var client = new HttpClient();
     var queryString = HttpUtility.ParseQueryString(string.Empty);

     client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", "{subscription
key}");
     queryString["ContentType"] = "Image";
     queryString["ContentId"] = "{string}";
     queryString["WorkflowName"] = "{string}";
     queryString["CallBackEndpoint"] = "{string}";
     var uri =
"https://westus.api.cognitive.microsoft.com/contentmoderator/review/v1.0/teams/{teamName
}/jobs?"
        + queryString;

      HttpResponseMessage response;
      byte[] byteData = Encoding.UTF8.GetBytes("{body}");

      using (var content = new ByteArrayContent(byteData))
      {
          content.Headers.ContentType = new MediaTypeHeaderValue("< your content type,
i.e. application/json >");
          response = await client.PostAsync(uri, content);
      }
```

Job Completion CallBack Sample

```
{
"JobId": "",
"ReviewId": “…",
"WorkFlowId": "default",
"Status": “…",
"ContentType": "Image",
"ContentId": "",
"CallBackType": "Job",
"Metadata": {
"adultscore": "0.xxx",
"a": "False",
"racyscore": "0.xxx",
"r": "True"
} }
```

- ◀ The Id of the Job submitted
- ◀ If the Job resulted in a Review
- ◀ Workflow Id used
- ◀ Complete, InProgress or Error
- ◀ Usually set to Image
- ◀ Indicates the Content Id
- ◀ Set to Job
- ◀ Whether the image contains Adult content
- ◀ Whether the image contains Racy content

Review Completion CallBack Sample

```
{ "ReviewId": "",
"ModifiedOn": "2016-10-
11T22:36:32.9934851Z",
"ModifiedBy": "",
"CallBackType": "Review",
"ContentId": "",
"Metadata": {
"adultscore": "0.xxx", "a": "False",
"racyscore": "0.xxx",
"r": "True"
},
"ReviewerResultTags": {
"a": "False",
"r": "True"
```

◂ The Id of Review

◂ DateTime modified

◂ Who modified the Job

◂ Set to Review

◂ Indicates the Content Id

◂ Adult Score and Boolean Tag

◂ Racy Score and Boolean Tag

◂ Adult and Racy values set by the Reviewer

Job API Methods: Get

```
curl -v -X GET

ttps://[location].api.cognitive.microsoft.com/contentmodera
tor/review/v1.0/teams/{teamName}/jobs/{JobId}"

-H "Content-Type: application/json"

-H "Ocp-Apim-Subscription-Key: {subscription key}“
```

Job API - Get Method

- location

- Ocp-Apim-Subscription-Key

Using Raw C#

```
public static async void MakeRequest()
{
    var client = new HttpClient();
    var queryString = HttpUtility.ParseQueryString(string.Empty);

    client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", "{subscription key}");

    var uri =

"https://westus.api.cognitive.microsoft.com/contentmoderator/review/v1.0/teams/{teamName
}/jobs/{JobId}?" +
         queryString;

    var response = await client.GetAsync(uri);
}
```

```
{
  "Id": "<Job Id>",
  "TeamName": "<Team Name>",
  "Status": "<Pending, Failed,
Completed>",
  "WorkflowId": "default",
  "CallBackEndpoint": "",
  "ReviewId": <Review Id if one is
created>,
  "ResultMetaData": [
   <Array of KeyValues with output from
Connectors>
]
```

◀ The Id of Job

◀ Team assigned to the Job

◀ Status of the Job

◀ Workflow Id used

◀ CallBackEndpoint

◀ Review Id if one exists

◀ Result Metadata

What Are Reviews?

Reviews are used to
create image or text reviews
for human moderation

Reviews Characteristics

Human-in-the-loop moderation

Review Tool is used

Follows automatic moderation

Workflow dependent

Final moderation step

They make use of the Review Tool

Review APIs Methods: Create

```
curl -v -X POST

"https://[location].api.cognitive.microsoft.com/contentmode
rator/review/v1.0/teams/{teamName}/reviews?subTeam={string}
"

-H "Content-Type: application/json"

-H "Ocp-Apim-Subscription-Key: {subscription key}"

--data-ascii "{body}"
```

Review API - Create Method

- location
- Ocp-Apim-Subscription-Key
- {body}

Using Raw C#

```
public static async void MakeRequest()
{
    var client = new HttpClient();
    var queryString = HttpUtility.ParseQueryString(string.Empty);

    client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", "{subscription key}");
    queryString["subTeam"] = "{string}";
    var uri =

"https://westus.api.cognitive.microsoft.com/contentmoderator/review/v1.0/teams/{teamName
}/reviews?"
        + queryString;

     HttpResponseMessage response;
     byte[] byteData = Encoding.UTF8.GetBytes("{body}");

     using (var content = new ByteArrayContent(byteData))
     {
         content.Headers.ContentType = new MediaTypeHeaderValue("< your content type,
i.e. application/json >");
         response = await client.PostAsync(uri, content);
     }
}
```

Demo

Review Tool

Demo

Using the Job and Review APIs with C#

Summary

Job & Review APIs

Review Tool

Human-in-the-loop Moderation with C#