Documentation of the REST API

## Using sample calls & expected results

This API is capable of returning results in both JSON and XML format. The content type of the response is dependent upon the accept headers provided by the request in the application.

For example, if you are trying the links below using Internet Explorer, your browser is likely to prompt you to save results as a JSON file. If you are using either Chrome or FireFox, the results are likely to display as XML.

Lichen: http://api.yourserver.org/api/SourceImages?sourceImageId=78B2B044-616D-E211-BE78-CC52AF888AB6.

By Alternate ID: http://api.yourserver.org/api/SourceImages?alternateId=904637.

Bryophyte: http://api.yourserver.org/api/SourceImages?sourceImageId=2C0E9CBA-626D-E211-BE78-CC52AF888AB6.

By Alternate ID: http://api.yourserver.org/api/SourceImages?alternateId=1417841.

# Create a new ParsingResult record by sourceUUID

POST: http://api.yourserver.org/api/parsingresults

Header:

Content-Length: 1079

Content-Type: application/json

{"parsingResultId":0,"sourceUUID":"38B89BA4-E9B9-4078-A796-FA5E38543A3E","dwcrecordedBy":"Richard C. Harris","dwcrecordNumber":"42164","dwcverbatimCoordinates":"41°11'N, 74°08'W","dwcverbatimEventDate":"4/19/1998","dwceventDate":"4/19/1998","dwcmunicipality":"","dwccounty":"Rockland","dwcstateProvince":"NEW YORK","dwccountry":"U.S.A.","aocrverbatimScientificName":"","dwcverbatimLocality":"Harriman State Park, along Woodtown Road West near dam at S end of Lake Sebago along Seven Lakes Drive","dwchabitat":"mixed hardwood-hemlock forest with granitic erratics.","dwcsubstrate":"on Trapelia placodioides Coppins & P. James","dwcverbatimElevation":"ca. 240 m","dwcidentifiedBy":"Daryl's Record","dwcdateIdentified":"","dwcverbatimLatitude":"41°11'N","dwcverbatimLongitude":"74°08'W","dwccatalogNumber":"1075759","aocrverbatimInstitution":"New York Botanical Garden","dwcdatasetName":"Lichens of New York State","dwcscientificName":"Polycoccum minutulum Kocourkova & F. Berger","dwcdecimalLatitude":"","dwcdecimalLongitude":"","dwcfieldNotes":"","dwcsex":"","createdByUserName":"Daryl"}

# Get a ParsingResult record by sourceUUID (JSON format)

GET: http://api.yourserver.org/api/parsingresults?identifier=38B89BA4-E9B9-4078-A796-FA5E38543A3E&createdByUserName=Daryl

Response Body:

HTTP/1.1 200 OK

{"parsingResultId":7,"fileNameIdentifier":null,"sourceUUID":"38b89ba4-e9b9-4078-a796-fa5e38543a3e","dwcrecordedBy":"Richard C. Harris","dwcrecordNumber":"42164","dwcverbatimCoordinates":"41°11'N, 74°08'W","dwcverbatimEventDate":"4/19/1998","dwceventDate":"4/19/1998","dwcmunicipality":"","dwccounty":"Rockland","dwcstateProvince":"NEW YORK","dwccountry":"U.S.A.","aocrverbatimScientificName":"","dwcverbatimLocality":"Harriman State Park, along Woodtown Road West near dam at S end of Lake Sebago along Seven Lakes Drive","dwchabitat":"mixed hardwood-hemlock forest with granitic erratics.","dwcsubstrate":"on Trapelia placodioides Coppins & P. James","dwcverbatimElevation":"ca. 240 m","dwcidentifiedBy":"Daryl's Record","dwcdateIdentified":"","dwcverbatimLatitude":"41°11'N","dwcverbatimLongitude":"74°08'W","dwccatalogNumber":"1075759","aocrverbatimInstitution":"New York Botanical Garden","dwcdatasetName":"Lichens of New York State","dwcscientificName":"Polycoccum minutulum Kocourkova & F. Berger","dwcdecimalLatitude":"","dwcdecimalLongitude":"","dwcfieldNotes":"","dwcsex":"","createdByUserName":"Daryl","createdUTCDateTime":"2013-02-14T18:34:59.213","ipAddress":"65.36.55.65"}

# Get CSV File Output by sourceUUID

GET: http://api.yourserver.org/api/parsingresultscsv?identifier=38b89ba4-e9b9-4078-a796-fa5e38543a3e&createdByUserName=Daryl

----

# Create a new ParsingResult record by FileName

POST: http://api.yourserver.org/api/parsingresults

Header:

Content-Length: 1079

Host: localhost:13514

Content-Type: application/json

{"parsingResultId":0,"fileNameIdentifier":"NY01075759\_lg.csv","dwcrecordedBy":"Richard C. Harris","dwcrecordNumber":"42164","dwcverbatimCoordinates":"41°11'N, 74°08'W","dwcverbatimEventDate":"4/19/1998","dwceventDate":"4/19/1998","dwcmunicipality":"","dwccounty":"Rockland","dwcstateProvince":"NEW YORK","dwccountry":"U.S.A.","aocrverbatimScientificName":"","dwcverbatimLocality":"Harriman State Park, along Woodtown Road West near dam at S end of Lake Sebago along Seven Lakes Drive","dwchabitat":"mixed hardwood-hemlock forest with granitic erratics.","dwcsubstrate":"on Trapelia placodioides Coppins & P. James","dwcverbatimElevation":"ca. 240 m","dwcidentifiedBy":"Bryan's Record","dwcdateIdentified":"","dwcverbatimLatitude":"41°11'N","dwcverbatimLongitude":"74°08'W","dwccatalogNumber":"1075759","aocrverbatimInstitution":"New York Botanical Garden","dwcdatasetName":"Lichens of New York State","dwcscientificName":"Polycoccum minutulum Kocourkova & F. Berger","dwcdecimalLatitude":"","dwcdecimalLongitude":"","dwcfieldNotes":"","dwcsex":"","createdByUserName":"Bryan"}

# Get a ParsingResult record by fileName (JSON format)

GET: http://api.yourserver.org/api/parsingresults?identifier=NY01075759\_lg.csv&createdByUserName=Bryan

Response Body:

HTTP/1.1 200 OK

{"parsingResultId":4,"fileNameIdentifier":"NY01075759\_lg.csv","sourceUUID":null,"dwcrecordedBy":"Richard C. Harris","dwcrecordNumber":"42164","dwcverbatimCoordinates":"41°11'N, 74°08'W","dwcverbatimEventDate":"4/19/1998","dwceventDate":"4/19/1998","dwcmunicipality":"","dwccounty":"Rockland","dwcstateProvince":"NEW YORK","dwccountry":"U.S.A.","aocrverbatimScientificName":"","dwcverbatimLocality":"Harriman State Park, along Woodtown Road West near dam at S end of Lake Sebago along Seven Lakes Drive","dwchabitat":"mixed hardwood-hemlock forest with granitic erratics.","dwcsubstrate":"on Trapelia placodioides Coppins & P. James","dwcverbatimElevation":"ca. 240 m","dwcidentifiedBy":"Bryan's Record","dwcdateIdentified":"","dwcverbatimLatitude":"41°11'N","dwcverbatimLongitude":"74°08'W","dwccatalogNumber":"1075759","aocrverbatimInstitution":"New York Botanical Garden","dwcdatasetName":"Lichens of New York State","dwcscientificName":"Polycoccum minutulum Kocourkova & F. Berger","dwcdecimalLatitude":"","dwcdecimalLongitude":"","dwcfieldNotes":"","dwcsex":"","createdByUserName":"Bryan","createdUTCDateTime":"2013-02-14T18:29:18.933","ipAddress":"65.36.55.65"}

# Get CSV File Output by file name

GET: http://api.yourserver.org/api/parsingresultscsv?identifier=NY01075759\_lg.csv&createdByUserName=Bryan

----

To access a test harness for part of the REST API, not covered by unit tests, go to: http://api.yourserver.org/ and click on the link next to item "2" that says, "Test Harness: Wanna play?" Or, just go straight to: http://api.yourserver.org?Test=basic

The REST API now supports receipt of OCR Requests, along with both callback and polling methods, and it will return back OCR results. This should be of great benefit to anyone that wants to take a crack at doing label parsing; especially if experimentation with multiple OCRs, including human-parsed output, is desired. A major benefit is that you do not have to deal with setting up your own OCR software to use this puppy. A drawback is that I see a lot of the OCR results are terrible.

Anyone with a browser can play with the example test harness and developers can view the source of the page as a reference and see how easy it is to use this REST API. Because this is a simulator for development against a known set of images, there is no need to actually upload the images themselves. Obviously, this would change on a "live" site with FTP upload or individual file upload. Please reference the attached image to clarify as I walk through some examples:

To use more than the few samples provide in the upper-right of the test harness page, download (click on the link) and use one or both of the CSV files provided at the top of the Web page:

200 Lichen: Good Lichen Test Candidates

More: 10,528 Test Candidates

## \* **Scenario A**: I know my file name is "NY01075759\_lg.jpg", but I do not know the UUID.

Enter a file name into the "Find by File" textbox and click the Search button. Results about that "Source Image" are displayed below including sourceImageId (UUID) of "89b2b044-616d-e211-be78-cc52af888ab6". That sourceImageId value can be used to create an OCR request.

REST call: http://api.yourserver.org/api/SourceImages?fileName=NY01075759\_lg.jpg

Response data: sourceImageId, filename, sourceImageRepositoryId, url, alternateId

## \* **Scenario B**: I know my sourceImageId (UUID) is "89b2b044-616d-e211-be78-cc52af888ab6" and I want information about it.

Enter the UUID value into the "Find by SourceImageId" textbox and click the Search button.

REST call: http://api.yourserver.org/api/SourceImages?sourceImageId=89b2b044-616d-e211-be78-cc52af888ab6

Response data: sourceImageId, filename, sourceImageRepositoryId, url, alternateId

## \* **Scenario C**: I want to simulate submission of an OCR request and use the polling method to get human-parsed results.

Enter the UUID value into the "Create OCR Request" textbox and click the "Create OCR Request" button. In about one second you should see the results appear below. This is actually an asynchronous operation, but behind-the-scenes I automated a click on the "Get OCR Results" button after a one-second delay. A developer would obviously have to write a bit of code to poll for results.

REST call (POST): http://api.yourserver.org/api/ImageProcessRequest

Request Body Parameters (JSON format):

{"sourceImageId":"89b2b044-616d-e211-be78-cc52af888ab6","processEngineId":"7","callbackUri":"","userName":"iDigBio"}

Response data (either JSON or XML format, depending upon your accept headers):

sourceImageProcessRequestId - Save this for subsequent polling requests or to correlate callback results

sourceImageId - echo of input parameter

processingEngineId - echo of input parameter

callbackUri - echo of input parameter

ipAddress - of the requesting machine. Future use for auditing usage, diagnosing problems, or refusing service if abuse or malicious behavior is detected

createdByUserName - echo of username input parameter. This could be used for rudimentary authorization, to assist in diagnosing client request submission issues,

or to aggregate/partition future submission of parsing results.

createdUTCDateTime - when the request was first received by the sever.

resultCreatedUTCDateTime - when the OCR engine finished its work) Note, under heavy volume, clients that track the difference between average

initial creation and result creation can make more intelligent decisions about how long to wait before attempting to poll for results.

resultValue - empty in the initial response to an OCR request submission and populated in the response during polling and callbacks (see below).

resultCallbackUri - empty in the initial response to an OCR request submission and populated in the response during polling and callbacks (see below).

\* **Scenario D**: I want to simulate submission of an OCR request and use the polling method to get ABBYY results.

Follow the steps in Scenario C above, then simply click on the "ABBYY" radio button. The difference is the "processEngineId" parameter will change from "7" (human-parsed) to "1" (ABBYY).

## \* **Scenario E**: I want to simulate submission of an OCR request WITH A CALLBACK (not polling):

Follow the steps in Scenario C above, but clear the "Callback Uri" textbox. The difference is the "callbackUri" parameter gets populated in the request body. The simulator is setup to make the callback the first time OCR results are requested (See Scenario F). Note, this would be done automatically in a true production environment.

REST call (POST): http://api.yourserver.org/api/ImageProcessRequest

Request Body Parameters (JSON format):

{"sourceImageId":"89b2b044-616d-e211-be78-cc52af888ab6","processEngineId":"7","callbackUri":"http://www.myserver.org/myCallbackPage","userName":"iDigBio"}

\* ScenarioF: Significant time has passed since I submitted my OCR request and my requesting server has not received the expected callback. I want to poll for results as a fallback.

Enter the "SourceImageProcessRequestId" received as a response to creating the original request into the "Poll for OCR Results" textbox. Click the "Get OCR Results" button.

REST call: http://api.yourserver.org/api/ImageProcessRequest/?sourceImageProcessRequestId=c96eb582-67b9-4723-a35d-0484cd7e3200

Response data: Same as Scenario "C" above, with these differences:

resultValue - now contains OCR results.

resultCallbackUri - contains the result of the callback notification.

The simulator intentionally only gives 2 seconds for the callback server to respond before timing out. Obviously, this would be configured differently in a non-test environment.

If the server/URL entered in the callback textbox is not setup to receive callbacks, a "404 Not Found" message is likely. However, if the callback is a valid page, a "200 OK" status

along with actual page content will result; although that isn't very useful outside of testing.

The API currently supports these three goals from the OCR SaaS Wiki page:

\* Accept incoming request and return a refId of the job.

\* Process the ocr with the available OCR engines (simulated)

\* Support calling endpoints

Developers, please use these values in the "processEngineId" parameter to retrieve results from the corresponding engine.

1 ABBYY

2 GOCR

3 OCRAD

4 OCROPUS

5 Tesseract

6 Xerox

7 Human

<http://www.scioqualis.com> ScioQualis.com has shared the source code for this REST API with iDigBio via an open source, MIT license.

~ Paul Schroeder & Robin Schroeder