Account API

# ES6 JavaScript

import ApiClient from "../ApiClient";

import Settings from '../model/Settings';

export default class AccountApi {

constructor(apiClient) {

this.apiClient = apiClient || ApiClient.instance;

}

getAccountSettings(account, password, callback) {

const postBody = null;

const pathParams = {};

const contentTypes = [];

const accepts = ['text/plain', 'application/json', 'text/json'];

const returnType = Settings;

const headerParams = {

'Account': account,

'Password': password

};

return this.apiClient.callApi(

'/api/v1/Account/GetAccountSettings',

'GET',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

}

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using RestSharp;

using IO.Swagger.Client;

using IO.Swagger.Model;

namespace IO.Swagger.Api

{

public interface IAccountApi : IApiAccessor

{

System.Threading.Tasks.Task<Settings> GetSettingsAsync(string account, string password);

System.Threading.Tasks.Task<ApiResponse<Settings>> GetSettingsWithHttpInfoAsync(string account, string password);

}

public partial class AccountApi : IAccountApi

{

private IO.Swagger.Client.ExceptionFactory \_exceptionFactory = (name, response) => null;

public AccountApi(String basePath)

{

this.Configuration = new IO.Swagger.Client.Configuration { BasePath = basePath };

ExceptionFactory = IO.Swagger.Client.Configuration.DefaultExceptionFactory;

}

public AccountApi()

{

this.Configuration = IO.Swagger.Client.Configuration.Default;

ExceptionFactory = IO.Swagger.Client.Configuration.DefaultExceptionFactory;

}

public AccountApi(IO.Swagger.Client.Configuration configuration = null)

{

if (configuration == null)

this.Configuration = IO.Swagger.Client.Configuration.Default;

else

this.Configuration = configuration;

ExceptionFactory = IO.Swagger.Client.Configuration.DefaultExceptionFactory;

}

public IO.Swagger.Client.Configuration Configuration { get; set; }

public IO.Swagger.Client.ExceptionFactory ExceptionFactory

{

get

{

if (\_exceptionFactory != null && \_exceptionFactory.GetInvocationList().Length > 1)

{

throw new InvalidOperationException("Multicast delegate for ExceptionFactory is unsupported.");

}

return \_exceptionFactory;

}

set { \_exceptionFactory = value; }

}

public async System.Threading.Tasks.Task<Settings> GetSettingsAsync(string account, string password)

{

ApiResponse<Settings> localVarResponse = await GetSettingsWithHttpInfoAsync(account, password);

return localVarResponse.Data;

}

public async System.Threading.Tasks.Task<ApiResponse<Settings>> GetSettingsWithHttpInfoAsync(string account, string password)

{

// verify the required parameter 'account' is set

if (account == null)

throw new ApiException(400, "Missing required parameter 'account' when calling AccountApi->ApiV1AccountGetAccountSettingsGet");

// verify the required parameter 'password' is set

if (password == null)

throw new ApiException(400, "Missing required parameter 'password' when calling AccountApi->ApiV1AccountGetAccountSettingsGet");

var localVarPath = "/api/v1/Account/GetAccountSettings";

var localVarPathParams = new Dictionary<String, String>();

var localVarQueryParams = new List<KeyValuePair<String, String>>();

var localVarHeaderParams = new Dictionary<String, String>(this.Configuration.DefaultHeader);

var localVarFormParams = new Dictionary<String, String>();

var localVarFileParams = new Dictionary<String, FileParameter>();

Object localVarPostBody = null;

// to determine the Content-Type header

String[] localVarHttpContentTypes = new String[] {

};

String localVarHttpContentType = this.Configuration.ApiClient.SelectHeaderContentType(localVarHttpContentTypes);

// to determine the Accept header

String[] localVarHttpHeaderAccepts = new String[] {

"text/plain",

"application/json",

"text/json"

};

String localVarHttpHeaderAccept = this.Configuration.ApiClient.SelectHeaderAccept(localVarHttpHeaderAccepts);

if (localVarHttpHeaderAccept != null)

localVarHeaderParams.Add("Accept", localVarHttpHeaderAccept);

if (account != null) localVarHeaderParams.Add("Account", this.Configuration.ApiClient.ParameterToString(account)); // header parameter

if (password != null) localVarHeaderParams.Add("Password", this.Configuration.ApiClient.ParameterToString(password)); // header parameter

// make the HTTP request

IRestResponse localVarResponse = (IRestResponse)await this.Configuration.ApiClient.CallApiAsync(localVarPath,

Method.GET, localVarQueryParams, localVarPostBody, localVarHeaderParams, localVarFormParams, localVarFileParams,

localVarPathParams, localVarHttpContentType);

int localVarStatusCode = (int)localVarResponse.StatusCode;

if (ExceptionFactory != null)

{

Exception exception = ExceptionFactory("ApiV1AccountGetAccountSettingsGet", localVarResponse);

if (exception != null) throw exception;

}

return new ApiResponse<Settings>(localVarStatusCode,

localVarResponse.Headers.ToDictionary(x => x.Name, x => string.Join(",", x.Value)),

(Settings)this.Configuration.ApiClient.Deserialize(localVarResponse, typeof(Settings)));

}

}

}

<?php

namespace Swagger\Client\Api;

use GuzzleHttp\Client;

use GuzzleHttp\ClientInterface;

use GuzzleHttp\Exception\RequestException;

use GuzzleHttp\Psr7\MultipartStream;

use GuzzleHttp\Psr7\Request;

use GuzzleHttp\RequestOptions;

use Swagger\Client\ApiException;

use Swagger\Client\Configuration;

use Swagger\Client\HeaderSelector;

use Swagger\Client\ObjectSerializer;

class AccountApi

{

protected $client;

protected $config;

protected $headerSelector;

public function \_\_construct(

ClientInterface $client = null,

Configuration $config = null,

HeaderSelector $selector = null

) {

$this->client = $client ?: new Client();

$this->config = $config ?: new Configuration();

$this->headerSelector = $selector ?: new HeaderSelector();

}

public function getConfig()

{

return $this->config;

}

public function apiV1AccountGetAccountSettingsGetAsync($account, $password)

{

return $this->apiV1AccountGetAccountSettingsGetAsyncWithHttpInfo($account, $password)

->then(

function ($response) {

return $response[0];

}

);

}

public function apiV1AccountGetAccountSettingsGetAsyncWithHttpInfo($account, $password)

{

$returnType = '\Swagger\Client\Model\Settings';

$request = $this->apiV1AccountGetAccountSettingsGetRequest($account, $password);

return $this->client

->sendAsync($request, $this->createHttpClientOption())

->then(

function ($response) use ($returnType) {

$responseBody = $response->getBody();

if ($returnType === '\SplFileObject') {

$content = $responseBody; //stream goes to serializer

} else {

$content = $responseBody->getContents();

if ($returnType !== 'string') {

$content = json\_decode($content);

}

}

return [

ObjectSerializer::deserialize($content, $returnType, []),

$response->getStatusCode(),

$response->getHeaders()

];

},

function ($exception) {

$response = $exception->getResponse();

$statusCode = $response->getStatusCode();

throw new ApiException(

sprintf(

'[%d] Error connecting to the API (%s)',

$statusCode,

$exception->getRequest()->getUri()

),

$statusCode,

$response->getHeaders(),

$response->getBody()

);

}

);

}

protected function apiV1AccountGetAccountSettingsGetRequest($account, $password)

{

// verify the required parameter 'account' is set

if ($account === null || (is\_array($account) && count($account) === 0)) {

throw new \InvalidArgumentException(

'Missing the required parameter $account when calling apiV1AccountGetAccountSettingsGet'

);

}

// verify the required parameter 'password' is set

if ($password === null || (is\_array($password) && count($password) === 0)) {

throw new \InvalidArgumentException(

'Missing the required parameter $password when calling apiV1AccountGetAccountSettingsGet'

);

}

$resourcePath = '/api/v1/Account/GetAccountSettings';

$formParams = [];

$queryParams = [];

$headerParams = [];

$httpBody = '';

$multipart = false;

// header params

if ($account !== null) {

$headerParams['Account'] = ObjectSerializer::toHeaderValue($account);

}

// header params

if ($password !== null) {

$headerParams['Password'] = ObjectSerializer::toHeaderValue($password);

}

// body params

$\_tempBody = null;

if ($multipart) {

$headers = $this->headerSelector->selectHeadersForMultipart(

['text/plain', 'application/json', 'text/json']

);

} else {

$headers = $this->headerSelector->selectHeaders(

['text/plain', 'application/json', 'text/json'],

[]

);

}

// for model (json/xml)

if (isset($\_tempBody)) {

// $\_tempBody is the method argument, if present

$httpBody = $\_tempBody;

// \stdClass has no \_\_toString(), so we should encode it manually

if ($httpBody instanceof \stdClass && $headers['Content-Type'] === 'application/json') {

$httpBody = \GuzzleHttp\json\_encode($httpBody);

}

} elseif (count($formParams) > 0) {

if ($multipart) {

$multipartContents = [];

foreach ($formParams as $formParamName => $formParamValue) {

$multipartContents[] = [

'name' => $formParamName,

'contents' => $formParamValue

];

}

// for HTTP post (form)

$httpBody = new MultipartStream($multipartContents);

} elseif ($headers['Content-Type'] === 'application/json') {

$httpBody = \GuzzleHttp\json\_encode($formParams);

} else {

// for HTTP post (form)

$httpBody = \GuzzleHttp\Psr7\build\_query($formParams);

}

}

$defaultHeaders = [];

if ($this->config->getUserAgent()) {

$defaultHeaders['User-Agent'] = $this->config->getUserAgent();

}

$headers = array\_merge(

$defaultHeaders,

$headerParams,

$headers

);

$query = \GuzzleHttp\Psr7\build\_query($queryParams);

return new Request(

'GET',

$this->config->getHost() . $resourcePath . ($query ? "?{$query}" : ''),

$headers,

$httpBody

);

}

protected function createHttpClientOption()

{

$options = [];

if ($this->config->getDebug()) {

$options[RequestOptions::DEBUG] = fopen($this->config->getDebugFile(), 'a');

if (!$options[RequestOptions::DEBUG]) {

throw new \RuntimeException('Failed to open the debug file: ' . $this->config->getDebugFile());

}

}

return $options;

}

}

# coding: utf-8

from \_\_future\_\_ import absolute\_import

import re # noqa: F401

# python 2 and python 3 compatibility library

import six

from swagger\_client.api\_client import ApiClient

class AccountApi(object):

def \_\_init\_\_(self, api\_client=None):

if api\_client is None:

api\_client = ApiClient()

self.api\_client = api\_client

def api\_v1\_account\_get\_account\_settings\_get(self, account, password, \*\*kwargs): # noqa: E501

kwargs['\_return\_http\_data\_only'] = True

if kwargs.get('async\_req'):

return self.api\_v1\_account\_get\_account\_settings\_get\_with\_http\_info(account, password, \*\*kwargs) # noqa: E501

else:

(data) = self.api\_v1\_account\_get\_account\_settings\_get\_with\_http\_info(account, password, \*\*kwargs) # noqa: E501

return data

def api\_v1\_account\_get\_account\_settings\_get\_with\_http\_info(self, account, password, \*\*kwargs): # noqa: E501

all\_params = ['account', 'password'] # noqa: E501

all\_params.append('async\_req')

all\_params.append('\_return\_http\_data\_only')

all\_params.append('\_preload\_content')

all\_params.append('\_request\_timeout')

params = locals()

for key, val in six.iteritems(params['kwargs']):

if key not in all\_params:

raise TypeError(

"Got an unexpected keyword argument '%s'"

" to method api\_v1\_account\_get\_account\_settings\_get" % key

)

params[key] = val

del params['kwargs']

# verify the required parameter 'account' is set

if ('account' not in params or

params['account'] is None):

raise ValueError("Missing the required parameter `account` when calling `api\_v1\_account\_get\_account\_settings\_get`") # noqa: E501

# verify the required parameter 'password' is set

if ('password' not in params or

params['password'] is None):

raise ValueError("Missing the required parameter `password` when calling `api\_v1\_account\_get\_account\_settings\_get`") # noqa: E501

collection\_formats = {}

path\_params = {}

query\_params = []

header\_params = {}

if 'account' in params:

header\_params['Account'] = params['account'] # noqa: E501

if 'password' in params:

header\_params['Password'] = params['password'] # noqa: E501

form\_params = []

local\_var\_files = {}

body\_params = None

# HTTP header `Accept`

header\_params['Accept'] = self.api\_client.select\_header\_accept(

['text/plain', 'application/json', 'text/json']) # noqa: E501

# Authentication setting

auth\_settings = [] # noqa: E501

return self.api\_client.call\_api(

'/api/v1/Account/GetAccountSettings', 'GET',

path\_params,

query\_params,

header\_params,

body=body\_params,

post\_params=form\_params,

files=local\_var\_files,

response\_type='Settings', # noqa: E501

auth\_settings=auth\_settings,

async\_req=params.get('async\_req'),

\_return\_http\_data\_only=params.get('\_return\_http\_data\_only'),

\_preload\_content=params.get('\_preload\_content', True),

\_request\_timeout=params.get('\_request\_timeout'),

collection\_formats=collection\_formats)