ES6 JavaScript

# API Client

import superagent from "superagent";

export default class ApiClient {

constructor() {

this.basePath = '/'.replace(/\/+$/, '');

this.authentications = {

type: 'basic',

username: '<username>',

};

this.defaultHeaders = {};

this.timeout = 60000;

this.cache = true;

this.enableCookies = false;

if (typeof window === 'undefined') {

this.agent = new superagent.agent();

}

this.requestAgent = null;

}

paramToString(param) {

if (param == undefined || param == null) {

return '';

}

if (param instanceof Date) {

return param.toJSON();

}

return param.toString();

}

buildUrl(path, pathParams) {

if (!path.match(/^\//)) {

path = '/' + path;

}

var url = this.basePath + path;

url = url.replace(/\{([\w-]+)\}/g, (fullMatch, key) => {

var value;

if (pathParams.hasOwnProperty(key)) {

value = this.paramToString(pathParams[key]);

} else {

value = fullMatch;

}

return encodeURIComponent(value);

});

return url;

}

isJsonMime(contentType) {

return Boolean(contentType != null && contentType.match(/^application\/json(;.\*)?$/i));

}

jsonPreferredMime(contentTypes) {

for (var i = 0; i < contentTypes.length; i++) {

if (this.isJsonMime(contentTypes[i])) {

return contentTypes[i];

}

}

return contentTypes[0];

}

normalizeParams(params) {

var newParams = {};

for (var key in params) {

if (params.hasOwnProperty(key) && params[key] != undefined && params[key] != null) {

var value = params[key];

if (Array.isArray(value)) {

newParams[key] = value;

} else {

newParams[key] = this.paramToString(value);

}

}

}

return newParams;

}

deserialize(response, returnType) {

if (response == null || returnType == null || response.status == 204) {

return null;

}

var data = response.body;

if (data == null || (typeof data === 'object' && typeof data.length === 'undefined' && !Object.keys(data).length)) {

data = response.text;

}

return ApiClient.convertToType(data, returnType);

}

callApi(path, httpMethod, pathParams,

headerParams, bodyParam, contentTypes, accepts,

returnType, callback) {

var url = this.buildUrl(path, pathParams);

var request = superagent(httpMethod, url);

// set header parameters

request.set(this.defaultHeaders).set(this.normalizeParams(headerParams));

// set requestAgent if it is set by user

if (this.requestAgent) {

request.agent(this.requestAgent);

}

// set request timeout

request.timeout(this.timeout);

var contentType = this.jsonPreferredMime(contentTypes);

if (contentType) {

// Issue with superagent and multipart/form-data (https://github.com/visionmedia/superagent/issues/746)

if(contentType != 'multipart/form-data') {

request.type(contentType);

}

} else if (!request.header['Content-Type']) {

request.type('application/json');

}

if (bodyParam) {

request.send(bodyParam);

}

var accept = this.jsonPreferredMime(accepts);

if (accept) {

request.accept(accept);

}

if (returnType === 'Blob') {

request.responseType('blob');

} else if (returnType === 'String') {

request.responseType('string');

}

// Attach previously saved cookies, if enabled

if (this.enableCookies){

if (typeof window === 'undefined') {

this.agent.attachCookies(request);

}

else {

request.withCredentials();

}

}

request.end((error, response) => {

if (callback) {

var data = null;

if (!error) {

try {

data = this.deserialize(response, returnType);

if (this.enableCookies && typeof window === 'undefined'){

this.agent.saveCookies(response);

}

} catch (err) {

error = err;

}

}

callback(error, data, response);

}

});

return request;

}

static parseDate(str) {

return new Date(str);

}

static convertToType(data, type) {

if (data === null || data === undefined)

return data

switch (type) {

case 'Boolean':

return Boolean(data);

case 'Integer':

return parseInt(data, 10);

case 'Number':

return parseFloat(data);

case 'String':

return String(data);

case 'Date':

return ApiClient.parseDate(String(data));

case 'Blob':

return data;

default:

if (type === Object) {

// generic object, return directly

return data;

} else if (typeof type === 'function') {

// for model type like: User

return type.constructFromObject(data);

} else if (Array.isArray(type)) {

// for array type like: ['String']

var itemType = type[0];

return data.map((item) => {

return ApiClient.convertToType(item, itemType);

});

} else if (typeof type === 'object') {

// for plain object type like: {'String': 'Integer'}

var keyType, valueType;

for (var k in type) {

if (type.hasOwnProperty(k)) {

keyType = k;

valueType = type[k];

break;

}

}

var result = {};

for (var d in data) {

if (data.hasOwnProperty(d)) {

var key = ApiClient.convertToType(d, keyType);

var value = ApiClient.convertToType(data[d], valueType);

result[key] = value;

}

}

return result;

} else {

// for unknown type, return the data directly

return data;

}

}

}

static constructFromObject(data, obj, itemType) {

if (Array.isArray(data)) {

for (var i = 0; i < data.length; i++) {

if (data.hasOwnProperty(i))

obj[i] = ApiClient.convertToType(data[i], itemType);

}

} else {

for (var k in data) {

if (data.hasOwnProperty(k))

obj[k] = ApiClient.convertToType(data[k], itemType);

}

}

}

}

ApiClient.instance = new ApiClient();

# Account API

## /api/v1/Account/GetAccountSettings

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

);

}

}

# Communication API

## /api/v1/Communication/EmailReceipt

import ApiClient from "../ApiClient";

import Response from '../model/Response';

export default class CommunicationApi {

constructor(apiClient) {

this.apiClient = apiClient || ApiClient.instance;

}

postEmailReceipt(account, password, opts, callback) {

opts = opts || {};

const postBody = opts['body'];

const pathParams = {};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = ['application/json', 'text/json', 'application/\_\*+json'];

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

const returnType = Response;

return this.apiClient.callApi(

'/api/v1/Communication/EmailReceipt',

'POST',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

}

# Reporting API

## /api/v1/Reporting/GetBatchesByDate/{batchDate}

import ApiClient from "../ApiClient";

import Batch from '../model/Batch';

import QueryTransaction from '../model/QueryTransaction';

import TerminalSettings from '../model/TerminalSettings';

export default class ReportingApi {

constructor(apiClient) {

this.apiClient = apiClient || ApiClient.instance;

}

getBatchesByDate(batchDate, account, password, callback) {

const postBody = null;

const pathParams = {

'batchDate': batchDate

};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = [];

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

const returnType = [Batch];

return this.apiClient.callApi(

`/api/v1/Reporting/GetBatchesByDate/${batchDate}`,

'GET',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

## /api/v1/Reporting/GetCheckTerminals

getTerminalsSettings(account, password, callback) {

const postBody = null;

const pathParams = {};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = [];

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

const returnType = [TerminalSettings];

return this.apiClient.callApi(

'/api/v1/Reporting/GetCheckTerminals',

'GET',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

## /api/v1/Reporting/GetCheckTerminalSettingsBySEC/{secCode}

getCheckTerminalSettingsBySecCode(secCode, account, password, callback) {

const postBody = null;

const pathParams = {

'secCode': secCode

};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = [];

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

const returnType = [TerminalSettings];

return this.apiClient.callApi(

`/api/v1/Reporting/GetCheckTerminalSettingsBySEC/${secCode}`,

'GET',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

## /api/v1/Reporting/GetCheckTerminalSettings/{terminalID}

getCheckTerminalSettingsById(terminalID, account, password, callback) {

const postBody = null;

const pathParams = {

'terminalID': terminalID

};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = [];

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

const returnType = [TerminalSettings];

return this.apiClient.callApi(

`/api/v1/Reporting/GetCheckTerminalSettings/${terminalID}`,

'GET',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

## /api/v1/Reporting/GetTransactionsByBatch/{batch}

getTransactionsByBatch(batch, account, password, callback) {

const postBody = null;

const pathParams = {

'batch': batch

};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = [];

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

const returnType = [QueryTransaction];

return this.apiClient.callApi(

`/api/v1/Reporting/GetTransactionsByBatch/${batch}`,

'GET',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

## /api/v1/Reporting/GetTransactionsByDate/{transactionDate}

getTransactionsByDate(transactionDate, account, password, callback) {

const postBody = null;

const pathParams = {

'transactionDate': transactionDate

};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = [];

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

const returnType = [QueryTransaction];

return this.apiClient.callApi(

`/api/v1/Reporting/GetTransactionsByDate/${transactionDate}`,

'GET',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

# Vault API

## /api/v1/Vault/GetVaultRecord/{reference}

import ApiClient from "../ApiClient";

import CheckTransaction from '../model/CheckTransaction';

import Record from '../model/Record';

import Response from '../model/Response';

import Transaction from '../model/Transaction';

export default class VaultApi {

constructor(apiClient) {

this.apiClient = apiClient || ApiClient.instance;

}

getVaultRecordReference(reference, account, password, callback) {

const postBody = null;

const pathParams = {

'reference': reference

};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = [];

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

const returnType = Record;

return this.apiClient.callApi(

`/api/v1/Vault/GetVaultRecord/${reference}`,

'GET',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

## /api/v1/Vault/SaveVault

saveVault(account, password, opts, callback) {

opts = opts || {};

const postBody = opts['body'];

const pathParams = {

};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = ['application/json', 'text/json', 'application/\_\*+json'];

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

const returnType = [Response];

return this.apiClient.callApi(

'/api/v1/Vault/SaveVault',

'POST',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

## /api/v1/Vault/SaveVaultCard/{vaultID}

saveVaultCardVaultId(vaultID, account, password, opts, callback) {

opts = opts || {};

const postBody = opts['body'];

const pathParams = {

'vaultID': vaultID

};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = ['application/json', 'text/json', 'application/\_\*+json'];

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

const returnType = Response;

return this.apiClient.callApi(

`/api/v1/Vault/SaveVaultCard/${vaultID}`,

'POST',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

## /api/v1/Vault/SearchVault

searchVault(account, password, opts, callback) {

opts = opts || {};

const postBody = opts['body'];

const pathParams = {

};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = ['application/json', 'text/json', 'application/\_\*+json'];

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

const returnType = [Record];

return this.apiClient.callApi(

'/api/v1/Vault/SearchVault',

'POST',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

## /api/v1/Vault/DeleteVaultCardByID/{vaultCardID}

deleteVaultCardByID(vaultCardID, account, password, callback) {

const postBody = null;

const pathParams = {

'vaultCardID': vaultCardID

};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = [];

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

const returnType = Response;

return this.apiClient.callApi(

`/api/v1/Vault/DeleteVaultCardByID/${vaultCardID}`,

'POST',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

## /api/v1/Vault/DeleteVaultCheckByID/{vaultCheckID}

deleteVaultCheckByID(vaultCheckID, account, password, callback) {

const postBody = null;

const pathParams = {

'vaultCheckID': vaultCheckID

};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = [];

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

const returnType = Response;

return this.apiClient.callApi(

`/api/v1/Vault/DeleteVaultCheckByID/${vaultCheckID}`,

'POST',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

## /api/v1/Vault/SubmitCheckWithVaultCheckID/{vaultCheckID}

submitCheckWithVaultCheckId(vaultCheckID, account, password, opts, callback) {

opts = opts || {};

const postBody = opts['body'];

const pathParams = {

'vaultCheckID': vaultCheckID

};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = ['application/json', 'text/json', 'application/\_\*+json'];

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

const returnType = CheckTransaction;

return this.apiClient.callApi(

'/api/v1/Vault/SubmitCheckWithVaultCheckID/{vaultCheckID}',

'POST',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

## /api/v1/Vault/SubmitWithVaultCardID/{vaultCardID}

submitWithVaultCardId(vaultCardID, account, password, opts, callback) {

opts = opts || {};

const postBody = opts['body'];

const pathParams = {

'vaultCardID': vaultCardID

};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = ['application/json', 'text/json', 'application/\_\*+json'];

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

const returnType = Transaction;

return this.apiClient.callApi(

'/api/v1/Vault/SubmitWithVaultCardID/{vaultCardID}',

'POST',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

# VirtualTerminal API

## /api/v1/VirtualTerminal/Submit

import ApiClient from "../ApiClient";

import CheckTransaction from '../model/CheckTransaction';

import QueryTransaction from '../model/QueryTransaction';

import Response from '../model/Response';

import Transaction from '../model/Transaction';

export default class VirtualTerminalApi {

constructor(apiClient) {

this.apiClient = apiClient || ApiClient.instance;

}

terminalSubmit(account, password, opts, callback) {

opts = opts || {};

const postBody = opts['body'];

const pathParams = {};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = ['application/json', 'text/json', 'application/\_\*+json'];

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

const returnType = Transaction;

return this.apiClient.callApi(

'/api/v1/VirtualTerminal/Submit',

'POST',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

## /api/v1/VirtualTerminal/SubmitCheck

terminalSubmitCheck(account, password, opts, callback) {

opts = opts || {};

const postBody = opts['body'];

const pathParams = {};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = ['application/json', 'text/json', 'application/\_\*+json'];

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

const returnType = CheckTransaction;

return this.apiClient.callApi(

'/api/v1/VirtualTerminal/SubmitCheck',

'POST',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

## /api/v1/VirtualTerminal/Mark/{transactionID}

markTransactionId(transactionID, account, password, callback) {

const postBody = null;

const pathParams = {

'transactionID': transactionID

};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = [];

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

const returnType = Response;

return this.apiClient.callApi(

`/api/v1/VirtualTerminal/Mark/${transactionID}`,

'POST',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

## /api/v1/VirtualTerminal/MarkTransactions

markTransactions(account, password, opts, callback) {

opts = opts || {};

const postBody = opts['body'];

const pathParams = {};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = ['application/json', 'text/json', 'application/\_\*+json'];

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

const returnType = Response;

return this.apiClient.callApi(

'/api/v1/VirtualTerminal/MarkTransactions',

'POST',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

## /api/v1/VirtualTerminal/Query

terminalQuery(account, password, opts, callback) {

opts = opts || {};

const postBody = opts['body'];

const pathParams = {};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = ['application/json', 'text/json', 'application/\_\*+json'];

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

const returnType = QueryTransaction;

return this.apiClient.callApi(

'/api/v1/VirtualTerminal/Query',

'POST',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}

## /api/v1/VirtualTerminal/UpdateTransactionInfo/{transactionID}

terminalUpdateTransactionInfo(transactionID, account, password, opts, callback) {

opts = opts || {};

const postBody = opts['body'];

const pathParams = {

'transactionID': transactionID

};

const headerParams = {

'Account': account,

'Password': password

};

const contentTypes = ['application/json', 'text/json', 'application/\_\*+json'];

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

const returnType = Response;

return this.apiClient.callApi(

`/api/v1/VirtualTerminal/UpdateTransactionInfo/${transactionID}`,

'POST',

pathParams,

headerParams,

postBody,

contentTypes,

accepts,

returnType,

callback

);

}