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;

}

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

);

}

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

);

}

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

);

}

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

);

}

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

);

}

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

);

}

}

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 IVirtualTerminalApi : IApiAccessor

{

System.Threading.Tasks.Task<Response> ApiV1VirtualTerminalMarkTransactionIDPostAsync(string transactionID, string account, string password);

System.Threading.Tasks.Task<ApiResponse<Response>> ApiV1VirtualTerminalMarkTransactionIDPostAsyncWithHttpInfo(string transactionID, string account, string password);

System.Threading.Tasks.Task<Response> ApiV1VirtualTerminalMarkTransactionsPostAsync(string account, string password, List<string> body = null);

System.Threading.Tasks.Task<ApiResponse<Response>> ApiV1VirtualTerminalMarkTransactionsPostAsyncWithHttpInfo(string account, string password, List<string> body = null);

System.Threading.Tasks.Task<QueryTransaction> ApiV1VirtualTerminalQueryPostAsync(string account, string password, Transaction body = null);

System.Threading.Tasks.Task<ApiResponse<QueryTransaction>> ApiV1VirtualTerminalQueryPostAsyncWithHttpInfo(string account, string password, Transaction body = null);

System.Threading.Tasks.Task<CheckTransaction> ApiV1VirtualTerminalSubmitCheckPostAsync(string account, string password, CheckTransaction body = null);

System.Threading.Tasks.Task<ApiResponse<CheckTransaction>> ApiV1VirtualTerminalSubmitCheckPostAsyncWithHttpInfo(string account, string password, CheckTransaction body = null);

System.Threading.Tasks.Task<Transaction> ApiV1VirtualTerminalSubmitPostAsync(string account, string password, Transaction body = null);

System.Threading.Tasks.Task<ApiResponse<Transaction>> ApiV1VirtualTerminalSubmitPostAsyncWithHttpInfo(string account, string password, Transaction body = null);

System.Threading.Tasks.Task<Response> ApiV1VirtualTerminalUpdateTransactionInfoTransactionIDPostAsync(string account, string password, string transactionID, UpdateTransactionInfo body = null);

System.Threading.Tasks.Task<ApiResponse<Response>> ApiV1VirtualTerminalUpdateTransactionInfoTransactionIDPostAsyncWithHttpInfo(string account, string password, string transactionID, UpdateTransactionInfo body = null);

}

public partial class VirtualTerminalApi : IVirtualTerminalApi

{

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

public VirtualTerminalApi(String basePath)

{

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

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

}

public VirtualTerminalApi()

{

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

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

}

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

{

if (configuration == null) // use the default one in Configuration

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<Response> ApiV1VirtualTerminalMarkTransactionIDPostAsync(string transactionID, string account, string password)

{

ApiResponse<Response> localVarResponse = await ApiV1VirtualTerminalMarkTransactionIDPostAsyncWithHttpInfo(transactionID, account, password);

return localVarResponse.Data;

}

public async System.Threading.Tasks.Task<ApiResponse<Response>> ApiV1VirtualTerminalMarkTransactionIDPostAsyncWithHttpInfo(string transactionID, string account, string password)

{

// verify the required parameter 'transactionID' is set

if (transactionID == null)

throw new ApiException(400, "Missing required parameter 'transactionID' when calling VirtualTerminalApi->ApiV1VirtualTerminalMarkTransactionIDPost");

// verify the required parameter 'account' is set

if (account == null)

throw new ApiException(400, "Missing required parameter 'account' when calling VirtualTerminalApi->ApiV1VirtualTerminalMarkTransactionIDPost");

// verify the required parameter 'password' is set

if (password == null)

throw new ApiException(400, "Missing required parameter 'password' when calling VirtualTerminalApi->ApiV1VirtualTerminalMarkTransactionIDPost");

var localVarPath = "/api/v1/VirtualTerminal/Mark/{transactionID}";

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 (transactionID != null) localVarPathParams.Add("transactionID", this.Configuration.ApiClient.ParameterToString(transactionID)); // path parameter

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.POST, localVarQueryParams, localVarPostBody, localVarHeaderParams, localVarFormParams, localVarFileParams,

localVarPathParams, localVarHttpContentType);

int localVarStatusCode = (int)localVarResponse.StatusCode;

if (ExceptionFactory != null)

{

Exception exception = ExceptionFactory("ApiV1VirtualTerminalMarkTransactionIDPost", localVarResponse);

if (exception != null) throw exception;

}

return new ApiResponse<Response>(localVarStatusCode,

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

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

}

public async System.Threading.Tasks.Task<Response> ApiV1VirtualTerminalMarkTransactionsPostAsync(string account, string password, List<string> body = null)

{

ApiResponse<Response> localVarResponse = await ApiV1VirtualTerminalMarkTransactionsPostAsyncWithHttpInfo(account, password, body);

return localVarResponse.Data;

}

public async System.Threading.Tasks.Task<ApiResponse<Response>> ApiV1VirtualTerminalMarkTransactionsPostAsyncWithHttpInfo(string account, string password, List<string> body = null)

{

// verify the required parameter 'account' is set

if (account == null)

throw new ApiException(400, "Missing required parameter 'account' when calling VirtualTerminalApi->ApiV1VirtualTerminalMarkTransactionsPost");

// verify the required parameter 'password' is set

if (password == null)

throw new ApiException(400, "Missing required parameter 'password' when calling VirtualTerminalApi->ApiV1VirtualTerminalMarkTransactionsPost");

var localVarPath = "/api/v1/VirtualTerminal/MarkTransactions";

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[] {

"application/json",

"text/json",

"application/\_\*+json"

};

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

if (body != null && body.GetType() != typeof(byte[]))

{

localVarPostBody = this.Configuration.ApiClient.Serialize(body); // http body (model) parameter

}

else

{

localVarPostBody = body; // byte array

}

// make the HTTP request

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

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

localVarPathParams, localVarHttpContentType);

int localVarStatusCode = (int)localVarResponse.StatusCode;

if (ExceptionFactory != null)

{

Exception exception = ExceptionFactory("ApiV1VirtualTerminalMarkTransactionsPost", localVarResponse);

if (exception != null) throw exception;

}

return new ApiResponse<Response>(localVarStatusCode,

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

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

}

public async System.Threading.Tasks.Task<QueryTransaction> ApiV1VirtualTerminalQueryPostAsync(string account, string password, Transaction body = null)

{

ApiResponse<QueryTransaction> localVarResponse = await ApiV1VirtualTerminalQueryPostAsyncWithHttpInfo(account, password, body);

return localVarResponse.Data;

}

public async System.Threading.Tasks.Task<ApiResponse<QueryTransaction>> ApiV1VirtualTerminalQueryPostAsyncWithHttpInfo(string account, string password, Transaction body = null)

{

// verify the required parameter 'account' is set

if (account == null)

throw new ApiException(400, "Missing required parameter 'account' when calling VirtualTerminalApi->ApiV1VirtualTerminalQueryPost");

// verify the required parameter 'password' is set

if (password == null)

throw new ApiException(400, "Missing required parameter 'password' when calling VirtualTerminalApi->ApiV1VirtualTerminalQueryPost");

var localVarPath = "/api/v1/VirtualTerminal/Query";

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[] {

"application/json",

"text/json",

"application/\_\*+json"

};

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

if (body != null && body.GetType() != typeof(byte[]))

{

localVarPostBody = this.Configuration.ApiClient.Serialize(body); // http body (model) parameter

}

else

{

localVarPostBody = body; // byte array

}

// make the HTTP request

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

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

localVarPathParams, localVarHttpContentType);

int localVarStatusCode = (int)localVarResponse.StatusCode;

if (ExceptionFactory != null)

{

Exception exception = ExceptionFactory("ApiV1VirtualTerminalQueryPost", localVarResponse);

if (exception != null) throw exception;

}

return new ApiResponse<QueryTransaction>(localVarStatusCode,

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

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

}

public async System.Threading.Tasks.Task<CheckTransaction> ApiV1VirtualTerminalSubmitCheckPostAsync(string account, string password, CheckTransaction body = null)

{

ApiResponse<CheckTransaction> localVarResponse = await ApiV1VirtualTerminalSubmitCheckPostAsyncWithHttpInfo(account, password, body);

return localVarResponse.Data;

}

public async System.Threading.Tasks.Task<ApiResponse<CheckTransaction>> ApiV1VirtualTerminalSubmitCheckPostAsyncWithHttpInfo(string account, string password, CheckTransaction body = null)

{

// verify the required parameter 'account' is set

if (account == null)

throw new ApiException(400, "Missing required parameter 'account' when calling VirtualTerminalApi->ApiV1VirtualTerminalSubmitCheckPost");

// verify the required parameter 'password' is set

if (password == null)

throw new ApiException(400, "Missing required parameter 'password' when calling VirtualTerminalApi->ApiV1VirtualTerminalSubmitCheckPost");

var localVarPath = "/api/v1/VirtualTerminal/SubmitCheck";

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[] {

"application/json",

"text/json",

"application/\_\*+json"

};

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

if (body != null && body.GetType() != typeof(byte[]))

{

localVarPostBody = this.Configuration.ApiClient.Serialize(body); // http body (model) parameter

}

else

{

localVarPostBody = body; // byte array

}

// make the HTTP request

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

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

localVarPathParams, localVarHttpContentType);

int localVarStatusCode = (int)localVarResponse.StatusCode;

if (ExceptionFactory != null)

{

Exception exception = ExceptionFactory("ApiV1VirtualTerminalSubmitCheckPost", localVarResponse);

if (exception != null) throw exception;

}

return new ApiResponse<CheckTransaction>(localVarStatusCode,

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

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

}

public async System.Threading.Tasks.Task<Transaction> ApiV1VirtualTerminalSubmitPostAsync(string account, string password, Transaction body = null)

{

ApiResponse<Transaction> localVarResponse = await ApiV1VirtualTerminalSubmitPostAsyncWithHttpInfo(account, password, body);

return localVarResponse.Data;

}

public async System.Threading.Tasks.Task<ApiResponse<Transaction>> ApiV1VirtualTerminalSubmitPostAsyncWithHttpInfo(string account, string password, Transaction body = null)

{

// verify the required parameter 'account' is set

if (account == null)

throw new ApiException(400, "Missing required parameter 'account' when calling VirtualTerminalApi->ApiV1VirtualTerminalSubmitPost");

// verify the required parameter 'password' is set

if (password == null)

throw new ApiException(400, "Missing required parameter 'password' when calling VirtualTerminalApi->ApiV1VirtualTerminalSubmitPost");

var localVarPath = "/api/v1/VirtualTerminal/Submit";

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[] {

"application/json",

"text/json",

"application/\_\*+json"

};

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

if (body != null && body.GetType() != typeof(byte[]))

{

localVarPostBody = this.Configuration.ApiClient.Serialize(body); // http body (model) parameter

}

else

{

localVarPostBody = body; // byte array

}

// make the HTTP request

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

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

localVarPathParams, localVarHttpContentType);

int localVarStatusCode = (int)localVarResponse.StatusCode;

if (ExceptionFactory != null)

{

Exception exception = ExceptionFactory("ApiV1VirtualTerminalSubmitPost", localVarResponse);

if (exception != null) throw exception;

}

return new ApiResponse<Transaction>(localVarStatusCode,

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

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

}

public async System.Threading.Tasks.Task<Response> ApiV1VirtualTerminalUpdateTransactionInfoTransactionIDPostAsync(string account, string password, string transactionID, UpdateTransactionInfo body = null)

{

ApiResponse<Response> localVarResponse = await ApiV1VirtualTerminalUpdateTransactionInfoTransactionIDPostAsyncWithHttpInfo(account, password, transactionID, body);

return localVarResponse.Data;

}

public async System.Threading.Tasks.Task<ApiResponse<Response>> ApiV1VirtualTerminalUpdateTransactionInfoTransactionIDPostAsyncWithHttpInfo(string account, string password, string transactionID, UpdateTransactionInfo body = null)

{

// verify the required parameter 'account' is set

if (account == null)

throw new ApiException(400, "Missing required parameter 'account' when calling VirtualTerminalApi->ApiV1VirtualTerminalUpdateTransactionInfoTransactionIDPost");

// verify the required parameter 'password' is set

if (password == null)

throw new ApiException(400, "Missing required parameter 'password' when calling VirtualTerminalApi->ApiV1VirtualTerminalUpdateTransactionInfoTransactionIDPost");

// verify the required parameter 'transactionID' is set

if (transactionID == null)

throw new ApiException(400, "Missing required parameter 'transactionID' when calling VirtualTerminalApi->ApiV1VirtualTerminalUpdateTransactionInfoTransactionIDPost");

var localVarPath = "/api/v1/VirtualTerminal/UpdateTransactionInfo/{transactionID}";

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[] {

"application/json",

"text/json",

"application/\_\*+json"

};

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 (transactionID != null) localVarPathParams.Add("transactionID", this.Configuration.ApiClient.ParameterToString(transactionID)); // path parameter

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

if (body != null && body.GetType() != typeof(byte[]))

{

localVarPostBody = this.Configuration.ApiClient.Serialize(body); // http body (model) parameter

}

else

{

localVarPostBody = body; // byte array

}

// make the HTTP request

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

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

localVarPathParams, localVarHttpContentType);

int localVarStatusCode = (int)localVarResponse.StatusCode;

if (ExceptionFactory != null)

{

Exception exception = ExceptionFactory("ApiV1VirtualTerminalUpdateTransactionInfoTransactionIDPost", localVarResponse);

if (exception != null) throw exception;

}

return new ApiResponse<Response>(localVarStatusCode,

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

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

}

}

}