AskAmex-Functions

ASKAMEX FUNCTIONS

AskAmex functions contains all the functions that make an API call request to the external API.Here we have 2 types of functions namely SOR and business based on need of the clients.

We have API's which are used for getting the transactions information of user, card offers available to user, payment history and many other details.

All the configs that are mentioned in the **askamex-functions** are defined in the **askamex-config-dev** code repository.

Here is the swagger documentation links for various functions which are part of askamex-functions

- https://github.aexp.com/pages/amex-eng/global-fins_global-fins-documents/myca_swagger.html#operation/getTransactionsV1
- https://explorer.aexp.com/?&redirects=2
- AskAmex Automation API Service Layer Documentation

Whenever we hit the API of Ask-Amex functions,the functions that execute irrespective of the clients request are namely remote_function_handler and local_function_handler.

remote_function_handler

This function checks whether the Authorization headers are present in the request headers or not.If authorization headers are present,then it makes a call to local function handler.

If the request headers does not contain the authorisation headers,then the response of the API will be the error message with status code 401.

remote_function_handler

```
@lazy_load_config(app_name=constants.APP_NAME)
async def remote_function_handler(request):
    request_header = request.headers
    auth = request_header.get('Authorization')
    if auth is None or str(auth.split(' ')[1]) != config.get('aaa.functions.app.id'):
        log.info(**{'correlation_id': correlation_id(request_header),
                    'app_name': constants.APP_NAME,
                    'event_name': 'Authorization',
                    'action': f'{constants.ACTION_END} Authorization',
                    'desc': 'exiting_execute',
                    'result': constants.LOG_FAILURE_RESULT,
                    'reason': 'Authorization failed',
                    'request_headers': str(request_header),
                    'app_id': config.get('aaa.functions.app.id')
                    })
        return web.Response(text=json.dumps({'error': constants.ERROR_MESSAGE}),
                            content_type='application/json',
                            status=401)
    else:
        function_response = await local_function_handler(request_header,
                                                         await request.json(), request.match_info.get
('function_name'))
        return web.Response(text=json.dumps(function_response[0]),
                            content_type='application/json',
                            status=function_response[1])
```

local_function_handler

Local function handler function directs the control to the actual function the client has requested.For example,if client has made a post call to get_account_transaction.Then this is the core part which routes the controller to execute function get_account_transaction function.Here you can also see the implementation of this function in source code block.

imported_fn -this variable is assigned with the value of the function file the user or client has actually requested.

faas_function-execute function in the imported function file

function_response-here we are calling the execute function with the request headers and request body

If the function response returns a status code 200,it sends a success message or else it sends a error message as response to request

Here why do we have remote_funtion_handler and local_function_handler separately because we can track easily where the function has failed either in the authorization or during the execution of the requested function.

local function handler

```
async def local_function_handler(request_header, request_body, function_name):
    imported_fn = importlib.import_module('aaa.functions.' + function_name)
    faas_function = getattr(imported_fn, 'execute')
    start_time = datetime.now()
    log.info(**{'correlation_id': correlation_id(request_header),
                'app_name': constants.APP_NAME,
                'event_name': function_name,
                'action': f'{constants.ACTION_START} {function_name}',
                'desc': 'entering_execute',
                'result': constants.LOG_SUCCESS_RESULT,
                'reason': constants.LOG_SUCCESS_REASON,
                'request_headers': str(request_header),
                'request_payload': str(request_body)
                })
    function_response = await faas_function(request_header, request_body)
    print("Faas function executed:function proxy")
    print(function_response[1])
    if function_response[1] == int(constants.RESPONSE_CODE_200):
        log.info(**{'correlation_id': correlation_id(request_header),
                    'app_name': constants.APP_NAME,
                    'event_name': function_name,
                    'action': f'{constants.ACTION_END} {function_name}',
                    'desc': 'exiting_execute',
                    'result': constants.LOG_SUCCESS_RESULT,
                    'reason': constants.LOG_SUCCESS_REASON,
                    'request_payload': str(function_response[0]),
                    'response_time': get_milli_seconds(start_time)
                    })
    else:
        log.error(**{'correlation_id': correlation_id(request_header),
                     'app_name': constants.APP_NAME,
                     'event_name': function_name,
                     'action': f'{constants.ACTION_END} {function_name}',
                     'desc': 'exiting_execute',
                     'result': constants.LOG_FAILURE_RESULT,
                     'reason': constants.LOG_FAILURE_REASON,
                     'request_payload': str(function_response[0]),
                     'response_time': get_milli_seconds(start_time)
                     })
    return function_response
```

SYSTEM OF RECORD FUNCTIONS

SOR functions echo back the response data from the external API.

One example of SOR function is get_acocunt_transaction_sor_v1

get_account_transaction_sor_v1

This function is used to send the user a list of transactions based on the account token.If the user requests transactions status as pending for that particular account token,then this API sends only pending transactions.If the user requests for posted transactions,this function sends the user a list of transactions with status as posted as response

sor_name:-This system of record name is used just for logging purpose.This is configured in aaa.config

sor_name

```
def sor_name():
    return config.get('aaa.functions.get.account.transaction.sor.name')
```

url()-The external url that we need to call from this function is returned in this function.

url

```
def url():
    return "http://mycaservicese2qonline.webqa.ipc.us.aexp.com:5555/financials/v1/transactions"
```

service_payload-In this function, we configure the request data that needs to be sent to the external API.

service_payload

```
def service_payload(arguments):
    payload = {
        "extended_details": arguments.get('extended_details', ''),
        "limit": int(arguments.get('limit', '10')),
        "status": arguments.get('status', ''),
        "description": arguments.get('description', '')
    }
    if arguments.get('status', '') != 'pending':
        payload['start_date'] = arguments.get('start_date', '')
        payload['end_date'] = arguments.get('end_date', '')

    return payload
```

headers-This headers function returns the headers that are required to make a request call to external API.

headers

```
def headers(correlation_id, a2a_token, account_token, user_jwt):
    return {
        constants.FINS_ACCOUNT_TOKENS: account_token,
        constants.FINS_CORRELATION_ID: correlation_id,
        constants.AUTHORIZATION_HEADER: f'Bearer {a2a_token}',
        constants.FINS_CLIENT_ID: config.get('aaa.functions.fins.client.id'),
        constants.FINS_USER_AUTHORIZATION: user_jwt
    }
```

execute():-This is function where all the logic related to the clients request is present.T

All the code is written in the try block,so if the code throws an exception then the except block will be executed and the response consists of an error message and status code with 500 .

Firstly,this functions checks whether the request body contains the required authorisation tokens or not.If the request does not contains the authorisation token,then it returns an error message with status code as 400.

Next,a post call is made to an external API with the headers and the service payload.

If this external API returns a success message with status code 200,then the response we got from the external API is sent to user as response body.

If the external API does not return a success message,then also the response from the external API is sent as response to user.Because,the SOR functions just echo the responses of external API.They do not modify the response data from the external API.

execute function of get_account_transaction_sor_v1

execute

```
async def execute(header, body):
    arguments, user, bot_memory = body.get('arguments'), body.get('user'), body.get('botMemory')
    try:
        correlation_id = header[constants.HEADER_CORRELATION_ID]
        user_jwt = header[constants.USER_AUTHORIZATION].split(' ')[1]
        account_token = arguments['accountToken']

        if bot_memory.get('conversationStringContext', {}).get('authorization_token'):
            a2a_token = bot_memory['conversationStringContext']['authorization_token']
        else:
            arguments['scope'] = ['/financials/v?/transactions**::get']
            a2a_bot_memory = copy.deepcopy(bot_memory)
            memory = {
                'arguments': arguments,
                'user': user,
                'botMemory': a2a_bot_memory
            }
            a2a_token_response = await local_function_handler(header, memory, 'get_a2a_token_v1')
            if a2a_token_response[1] == int(constants.RESPONSE_CODE_200):
                a2a_token = a2a_token_response[0]['botMemory']['conversationStringContext']
['authorization_token']
            else:
                return {'error': a2a_token_response[0]['statusMessage']}, int(constants.RESPONSE_CODE_400)
        response = await aio_http_get(correlation_id, sor_name(), url(),
                                      headers(correlation_id, a2a_token, account_token, user_jwt),
                                      timeout(), service_payload(arguments))
        if response[1] == int(constants.RESPONSE_CODE_200):
            return {'response': response[0]}, 200
        else:
            return {'response': response[0], 'accountToken': account_token}, int(response[1])
    except Exception as error:
        return {'error': str(error)}, int(constants.RESPONSE_CODE_500)
```

BUSINESS FUNCTIONS

Business functions hold the business logic based on the requesting clients

Here is an example of one business function **get_account_transaction_v1**

get_account_transaction_v1

This function business logic is to send the user a list of transactions based on the account token irrespective of their status i.e. either the transaction is present in pending state or posted state.

Functions such as sor_name,url,service_payload,headers are common and implemented the exactly similarly in both sor functions and business functions.Business functions also consists of response_builder function and other helper functions in addition.Response builder function is used to configure the response that needs to be sent to the clients based on the clients requests.

sor_name:-This system of record name is used just for logging purpose.This is configured in aaa.config.

url()-The external url that we need to call from this function is returned in this function.

service_payload-In this function, we configure the request data that needs to be sent to the external API.

headers-This headers function returns the headers that are required to make a request call to external API.

execute-

This execute function calls get_account_transaction_sor_v1 API twice with status as pending one time and status as posted the another time for every account token in the request body.If we get a status code as 200 from the API,then we append the responses from the posted_request body and pending request body.

If either of the API call to get_account_transaction_sor_v1 fails i.e. if either the request with status as pending or the request with status as posted fails,we send an error message to client with status code 500.

So,here in this business function you have observed the difference between SOR response and business function.With get_account_transaction_sor_v1 we can retrive only either posted or pending transactions.But in this get_account_transaction_v3, we can receive both types of transactions.

execute function in get_account_transaction_v1

execute function code

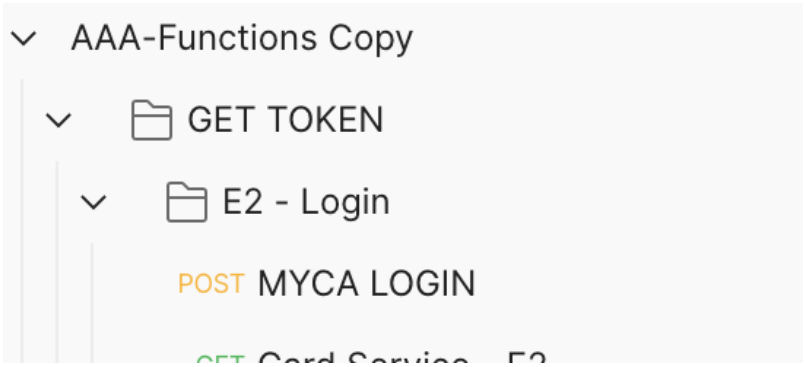
```
async def execute(header, body):
    try:
        arguments, user, bot_memory = body.get('arguments'), body.get('user'), body.get('botMemory')
        arguments['scope'] = ['/financials/v?/transactions**::get']
        posted_body = get_body_with_status(body, 'posted')
        await local_function_handler(header, posted_body, 'get_a2a_token_v1')
        pending_body = get_body_with_status(posted_body, 'pending')
        account_transactions = []

        for account_token in arguments['accountToken']:
            posted_body['arguments']['accountToken'] = account_token
            pending_body['arguments']['accountToken'] = account_token

            tasks = [
                local_function_handler(header, posted_body, 'get_account_transaction_sor_v1'),
                local_function_handler(header, pending_body, 'get_account_transaction_sor_v1')
            ]
            results = await asyncio.gather(*tasks)
            response_posted = results[0]
            response_pending = results[1]
            if response_posted[1] == int(constants.RESPONSE_CODE_200) \
                    and response_pending[1] == int(constants.RESPONSE_CODE_200):
                response_posted = create_transaction_list(response_posted, 'TRANSACTION_STATUS_POSTED')
                response_pending = create_transaction_list(response_pending, 'TRANSACTION_STATUS_PENDING')
                transactions_response = response_posted['transactions'] + response_pending['transactions']
                account_transactions.append(
                    get_transactions_list(arguments, transactions_response, account_token, 'success'))
            else:
                return {'status': 'failure',
                        'statusMessage': 'Our system is not responding',
                        'user': user,
                        'botMemory': bot_memory}, int(constants.RESPONSE_CODE_500)
        return response_builder(user, bot_memory, account_transactions, 200)
    except Exception as error:
        return {'status': 'failure',
                'statusMessage': str(error),
                'user': user,
                'botMemory': bot_memory}, int(constants.RESPONSE_CODE_400)
```

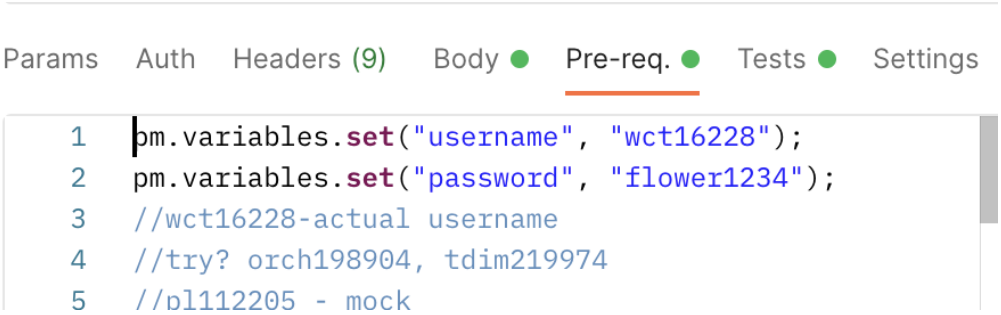
TESTING OF ASKAMEX-FUNCTIONS

To test the askamex-functions locally,you need to import postman collection of askmex-functions.

Firstly to send to any request to askamex-functions,you need to login using MYCA LOGIN,then the authorization headers are set and we can send any post,get requests to AAA-functions.

The MYCA LOGIN is present in postman collection.You need to set username and password of the test account you want to test in the pre-request scripts of the MYCA LOGIN





You also need to change the url of the function you want to test from {{aaa}}/aaa/functions/v1/read_account_limits to http://localhost:6000/aaa/functions/v1/read_account_limits

Here is an example of post request to get_account_transaction_sor_v1 API.

get_account_transaction_sor_v1

This is the request body for get_acocunt_transaction_sor_v1.You can change in the request body whether you want pending transactions or posted transactions.

request body

```
{
    "arguments": {
        "accountToken": "{{accountToken}}",
        "extended_details": "merchant,category,receipts",
        "status": "posted",
        "start_date": "2021-02-04",
        "end_date": "2021-05-20",
        "limit": "10"
    },
    "user": {},
    "botMemory": {
        "conversationStringContext": {
            "journey": "account_transaction"
        },
        "sessionStringContext": {}
    }
}
```

headers

KEY	VALUE
X-AXP-CorrelationId	{{$guid}}
Authorization	Auth {{faasappid}}
Content-Type	application/json

X-AXP-User-Authorization	User_JWT {{jwt}}

response body for get_acocunt_transaction_sor_v1

response_body


```
{
    "response": [
        {
            "total_transaction_count": 47,
            "status": "posted",
            "transactions": [
                {
                    "account_token": "Q019TGVHLSWCFCE",
                    "identifier": "320210970080172021",
                    "pan_last_five": "42000",
                    "charge_date": "2021-04-06",
                    "post_date": "2021-04-07",
                    "statement_end_date": "2021-04-12",
                    "amount": "-446.25",
                    "supplementary_digits": "00",
                    "reference_number": "320210970080172021",
                    "transaction_id": "0",
                    "type": "CREDIT",
                    "sub_type": "payment",
                    "event": "PAYMENT - THANK YOU",
                    "description": "MINDUE PAYMENT",
                    "extended_details": {
                        "merchant": {
                            "identifier": "3126722420",
                            "master_affiliated_identifier": "4450642665",
                            "chain_affiliated_identifier": "4450642632",
                            "name": "SPIEGEL-MAIL ORDER",
                            "address": {
                                "address_lines": [
                                    "3500 LACEY RD"
                                ],
                                "country_name": "UNITED STATES OF AMERICA (THE)",
                                "postal_code": "60515-5422",
                                "city": "DOWNERS GROVE",
                                "state": "IL"
                            },
                            "phone_number": "6309868800",
                            "merchant_url": "https://www.SPIEGEL-MAILORDER.com",
                            "additional_url": "https://www.SPIEGEL-MAILORDER/index.htm"
                        },
                        "category": {
                            "category_code": "C6",
                            "category_name": "Other",
                            "subcategory_code": "50",
                            "subcategory_name": "Miscellaneous"
                        }
                    }
                },
                {
                    "account_token": "Q019TGVHLSWCFCE",
                    "identifier": "320210713078651677",
                    "pan_last_five": "42000",
                    "charge_date": "2021-03-12",
                    "post_date": "2021-03-12",
                    "statement_end_date": "2021-03-12",
                    "amount": "-376.91",
                    "supplementary_digits": "00",
                    "reference_number": "320210713078651677",
                    "transaction_id": "0",
                    "type": "CREDIT",
                    "sub_type": "PURCHASE",
                    "event": "EARLY PAY DISCOUNT",
                    "description": "EARLY PAY DISCOUNT",
```

```
            "extended_details": {
                "category": {
                    "category_code": "C4",
                    "category_name": "Fees & Adjustments",
                    "subcategory_code": "27",
                    "subcategory_name": "Fees & Adjustments"
                }
            }
        },
        {
            "account_token": "Q019TGVHLSWCFCE",
            "identifier": "320210530138343745",
            "pan_last_five": "42000",
            "charge_date": "2021-02-21",
            "post_date": "2021-02-22",
            "statement_end_date": "2021-03-12",
            "amount": "98.25",
            "supplementary_digits": "00",
            "reference_number": "320210530138343745",
            "transaction_id": "983962151079",
            "type": "DEBIT",
            "sub_type": "PURCHASE",
            "event": "PURCHASE",
            "description": "STAPLES 00452       PHOENIX             AZ",
            "extended_details": {
                "merchant": {
                    "identifier": "6310805403",
                    "name": "STAPLES 452",
                    "address": {
                        "address_lines": [
                            "601 TROY SCHENECTADY RD"
                        ],
                        "country_name": "UNITED STATES",
                        "postal_code": "12110-2813",
                        "city": "LATHAM",
                        "state": "NY"
                    },
                    "phone_number": "555-555-5555",
                    "merchant_url": "https://www.STAPLES452.com",
                    "additional_url": "https://www.STAPLES452/index.htm"
                },
                "category": {
                    "category_code": "C6",
                    "category_name": "Other",
                    "subcategory_code": "50",
                    "subcategory_name": "Miscellaneous"
                }
            }
        },
        {
            "account_token": "Q019TGVHLSWCFCE",
            "identifier": "320210520120206992",
            "pan_last_five": "42000",
            "charge_date": "2021-02-21",
            "post_date": "2021-02-21",
            "statement_end_date": "2021-03-12",
            "amount": "47.11",
            "supplementary_digits": "00",
            "reference_number": "320210520120206992",
            "transaction_id": "1082123984061",
            "type": "DEBIT",
            "sub_type": "PURCHASE",
            "event": "PURCHASE",
            "description": "TST* CITY SQUIRE ALEPHOENIX             AZ",
            "extended_details": {
                "merchant": {
                    "identifier": "4310422142",
                    "name": "NEW MERCHANT"
                },
                "category": {
                    "category_code": "C6",
```

```
                "category_name": "Other",
                "subcategory_code": "50",
                "subcategory_name": "Miscellaneous"
            }
        }
    },
    {
        "account_token": "Q019TGVHLSWCFCE",
        "identifier": "320210510109644395",
        "pan_last_five": "42000",
        "charge_date": "2021-02-20",
        "post_date": "2021-02-20",
        "statement_end_date": "2021-03-12",
        "amount": "59.00",
        "supplementary_digits": "00",
        "reference_number": "320210510109644395",
        "transaction_id": "1078605972061",
        "type": "DEBIT",
        "sub_type": "PURCHASE",
        "event": "PURCHASE",
        "description": "9MILES ORDER#37669  PHOENIX             AZ",
        "extended_details": {
            "merchant": {
                "identifier": "2313409872",
                "name": "NEW MERCHANT"
            },
            "category": {
                "category_code": "C6",
                "category_name": "Other",
                "subcategory_code": "50",
                "subcategory_name": "Miscellaneous"
            }
        }
    },
    {
        "account_token": "Q019TGVHLSWCFCE",
        "identifier": "320210520123434395",
        "pan_last_five": "42000",
        "charge_date": "2021-02-20",
        "post_date": "2021-02-21",
        "statement_end_date": "2021-03-12",
        "amount": "17.17",
        "supplementary_digits": "00",
        "reference_number": "320210520123434395",
        "transaction_id": "8475355607876",
        "type": "DEBIT",
        "sub_type": "PURCHASE",
        "event": "PURCHASE",
        "description": "EXXONMOBIL 9736     PHOENIX             AZ",
        "extended_details": {
            "merchant": {
                "identifier": "3248400871",
                "master_affiliated_identifier": "1428473656",
                "chain_affiliated_identifier": "3248400293",
                "name": "EXXONMOBIL CAT OUTSIDE",
                "address": {
                    "address_lines": [
                        "CUST SVC 1 800 243-9966"
                    ],
                    "country_name": "UNITED STATES OF AMERICA (THE)",
                    "postal_code": "64141",
                    "city": "KANSAS CITY",
                    "state": "MO"
                },
                "phone_number": "555-555-5555",
                "merchant_url": "http://www.exxonmobilstations.com/",
                "additional_url": "http://corporate.exxonmobil.com/en/company/contact-us"
            },
            "category": {
                "category_code": "C6",
                "category_name": "Other",
```

```
            "subcategory_code": "50",
            "subcategory_name": "Miscellaneous"
        }
    }
},
{
    "account_token": "Q019TGVHLSWCFCE",
    "identifier": "320210520119515519",
    "pan_last_five": "42000",
    "charge_date": "2021-02-20",
    "post_date": "2021-02-21",
    "statement_end_date": "2021-03-12",
    "amount": "54.14",
    "supplementary_digits": "00",
    "reference_number": "320210520119515519",
    "transaction_id": "975403569075",
    "type": "DEBIT",
    "sub_type": "PURCHASE",
    "event": "PURCHASE",
    "description": "NISKAYUNA CO-OP 0000PHOENIX            AZ",
    "extended_details": {
        "merchant": {
            "identifier": "6312661655",
            "name": "NISKAYUNA CO-OP 0879917",
            "address": {
                "address_lines": [
                    "2227 NOTT ST"
                ],
                "country_name": "UNITED STATES",
                "postal_code": "12309-4326",
                "city": "SCHENECTADY",
                "state": "NY"
            },
            "phone_number": "555-555-5555",
            "merchant_url": "https://www.NISKAYUNACO-OP0879917.com",
            "additional_url": "https://www.NISKAYUNACO-OP0879917/index.htm"
        },
        "category": {
            "category_code": "C6",
            "category_name": "Other",
            "subcategory_code": "50",
            "subcategory_name": "Miscellaneous"
        }
    }
},
{
    "account_token": "Q019TGVHLSWCFCE",
    "identifier": "320210500098898290",
    "pan_last_five": "42000",
    "charge_date": "2021-02-19",
    "post_date": "2021-02-19",
    "statement_end_date": "2021-03-12",
    "amount": "9.99",
    "supplementary_digits": "00",
    "reference_number": "320210500098898290",
    "transaction_id": "969162693077",
    "type": "DEBIT",
    "sub_type": "PURCHASE",
    "event": "PURCHASE",
    "description": "GOOGLE*YOUTUBEPREMIUPHOENIX            AZ",
    "extended_details": {
        "merchant": {
            "identifier": "1047161427",
            "master_affiliated_identifier": "1040041436",
            "chain_affiliated_identifier": "1047161351",
            "name": "GOOGLE SERVICES",
            "address": {
                "address_lines": [
                    "1600 AMPHITHEATRE PKWY"
                ],
                "country_name": "UNITED STATES OF AMERICA (THE)",
```

```
                "postal_code": "94043-1351",
                "city": "MOUNTAIN VIEW",
                "state": "CA"
            },
            "phone_number": "8554925538",
            "merchant_url": "https://play.google.com/store",
            "additional_url": "https://support.google.com/googleplay#topic=3364671"
        },
        "category": {
            "category_code": "C6",
            "category_name": "Other",
            "subcategory_code": "50",
            "subcategory_name": "Miscellaneous"
        }
    }
},
{
    "account_token": "Q019TGVHLSWCFCE",
    "identifier": "3202105000085400357",
    "pan_last_five": "42000",
    "charge_date": "2021-02-19",
    "post_date": "2021-02-19",
    "statement_end_date": "2021-03-12",
    "amount": "-24933.14",
    "supplementary_digits": "00",
    "reference_number": "320210500085400357",
    "transaction_id": "0",
    "type": "CREDIT",
    "sub_type": "payment",
    "event": "PAYMENT - THANK YOU",
    "description": "ONLINE PAYMENT - THAPHOENIX             AZ"
},
{
    "account_token": "G1WRY4L9MN7TNEG",
    "identifier": "320210490079943355",
    "pan_last_five": "43040",
    "charge_date": "2021-02-18",
    "post_date": "2021-02-18",
    "statement_end_date": "2021-03-12",
    "amount": "56.15",
    "supplementary_digits": "04",
    "reference_number": "320210490079943355",
    "transaction_id": "8442514706379",
    "type": "DEBIT",
    "sub_type": "PURCHASE",
    "event": "PURCHASE",
    "description": "MAILCHIMP           PHOENIX             AZ",
    "extended_details": {
        "merchant": {
            "identifier": "4100250307",
            "name": "MAILCHIMP",
            "address": {
                "address_lines": [
                    "512 MEANS ST NW",
                    "STE 404"
                ],
                "country_name": "UNITED STATES",
                "postal_code": "30318-5788",
                "city": "ATLANTA",
                "state": "GA"
            },
            "phone_number": "555-555-5555",
            "merchant_url": "https://www.MAILCHIMP.com",
            "additional_url": "https://www.MAILCHIMP/index.htm"
        },
        "category": {
            "category_code": "C6",
            "category_name": "Other",
            "subcategory_code": "50",
            "subcategory_name": "Miscellaneous"
        }
```

```
                    }
                }
            ]
        }
    ]
}
```

get_account_transaction_v1

request_body

request_body

```
{
  "arguments": {
    "accountToken": [
      "{{accountToken}}"
    ],
    "extended_details": "merchant,category",
    "limit": "15",
    "start_date": "2021-07-15",
    "end_date": "2021-10-12"
  },
  "user": {},
  "botMemory": {
    "conversationStringContext": {
      "journey": "account_transaction"
    },
    "sessionStringContext": {}
  }
}
```

headers

KEY	VALUE
X-AXP-CorrelationId	{{$guid}}
Authorization	Auth {{faasappid}}
Content-Type	application/json
X-AXP-User-Authorization	User_JWT {{jwt}}

response_body

response


```
{
    "user": {},
    "botMemory": {
        "conversationStringContext": {
            "journey": "account_transaction",
            "transactions": [
                {
                    "accountToken": "Q019TGVHLSWCFCE",
                    "status": "success",
                    "transactions": [
                        {
                            "transaction_identifier": "320210970080172021",
                            "merchant_name": "MINDUE PAYMENT",
                            "amount": "-446.25",
                            "description": "MINDUE PAYMENT",
                            "transaction_date": "2021-04-06",
                            "statement_end_date": "2021-04-12",
                            "category": "",
                            "sub_category": "",
                            "flags": "",
                            "city": "",
                            "state": "",
                            "status": "TRANSACTION_STATUS_POSTED"
                        },
                        {
                            "transaction_identifier": "320210713078651677",
                            "merchant_name": "EARLY PAY DISCOUNT",
                            "amount": "-376.91",
                            "description": "EARLY PAY DISCOUNT",
                            "transaction_date": "2021-03-12",
                            "statement_end_date": "2021-03-12",
                            "category": "",
                            "sub_category": "",
                            "flags": "",
                            "city": "",
                            "state": "",
                            "status": "TRANSACTION_STATUS_POSTED"
                        }
                    ],
                    "total_transaction_amount": -823.16
                }
            ]
        },
        "sessionStringContext": {}
    },
    "status": "success",
    "statusMessage": "Data fetched successfully"
}
```