Collabothon API Usage

20 August 2018 16:29

Using the REST APIs

For the Collabothon 2018, two REST APIs are provided by Commerzbank for you to play around with. The **Accounts API** and **Customers API**.

You can use the **Accounts API** to:

- view a list of accounts
- view specific details of an account
- create or delete an account
- view balances of an account
- view a list of transactions to an account, as well as details to a specific transaction

The **Customers API** lets you:

- find customers by first/last name, address and date of birth
- view detailed information to a customer
- view accounts related to a customer
- view addresses, legitimation data and tax-IDs to a customer

We have created six persons for which you will be able to find information: *Ariane*, *Jürgen*, *Lea*, *Marc*, *René* and *Sabine*. Using the Customers API, you can go ahead and look for the their corresponding bank accounts.

Both APIs are accessible through the Internet, provided you authenticate yourself through your individual API Key. You will be provided with your key when the hacking begins. You through your terminal via cURL, or any given REST Client, whatever feels comfortable. To get a quick and easy understanding, we recommend you try using cURL (https://curl.haxx. The APIs are accessible at:

Accounts API:

URL: https://api-sandbox.commerzbank.com/accounts-api/v1-s

KeyId: <your keyId>

Customers API:

URL: https://api-sandbox.commerzbank.com/customers-api/v1-s

KeyId: <your keyId>

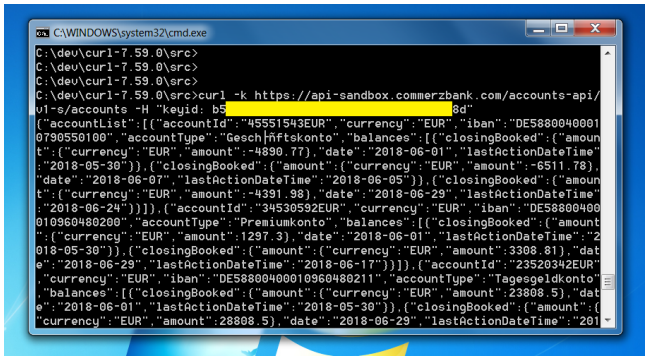
So you can start exploring the APIs by calling:

> curl -k https://api-sandbox.commerzbank.com/accounts-api/v1-s/accounts -H "keyid: <your keyId>"

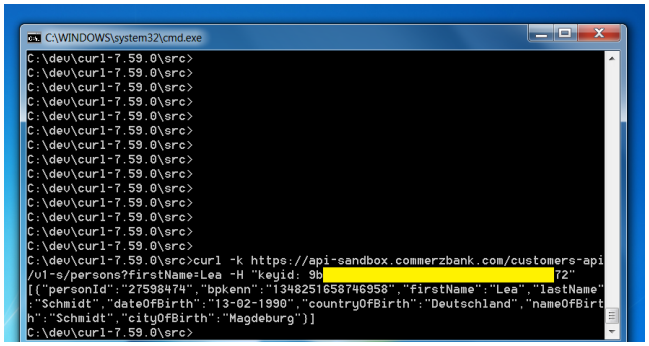
> curl -k https://api-sandbox.commerzbank.com/customers-api/v1-s/persons?firstName=Lea -H "keyid: <your keyId>"

If you prefer a graphical interface, try using one of the many REST Clients, for example Insomnia (https://insomnia.rest/). You can see examples using both variants here:

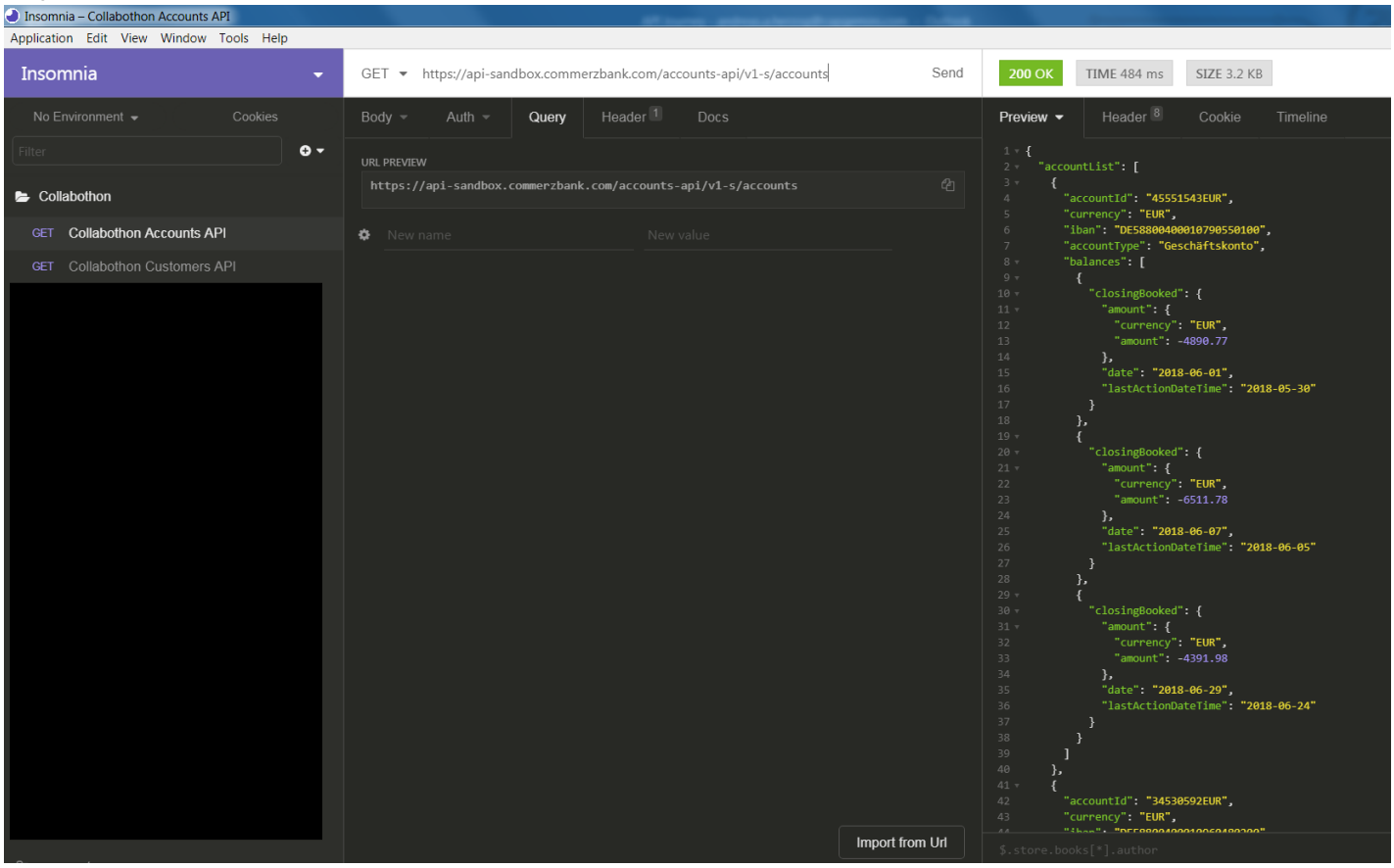
Using cURL to access the **Accounts API**:



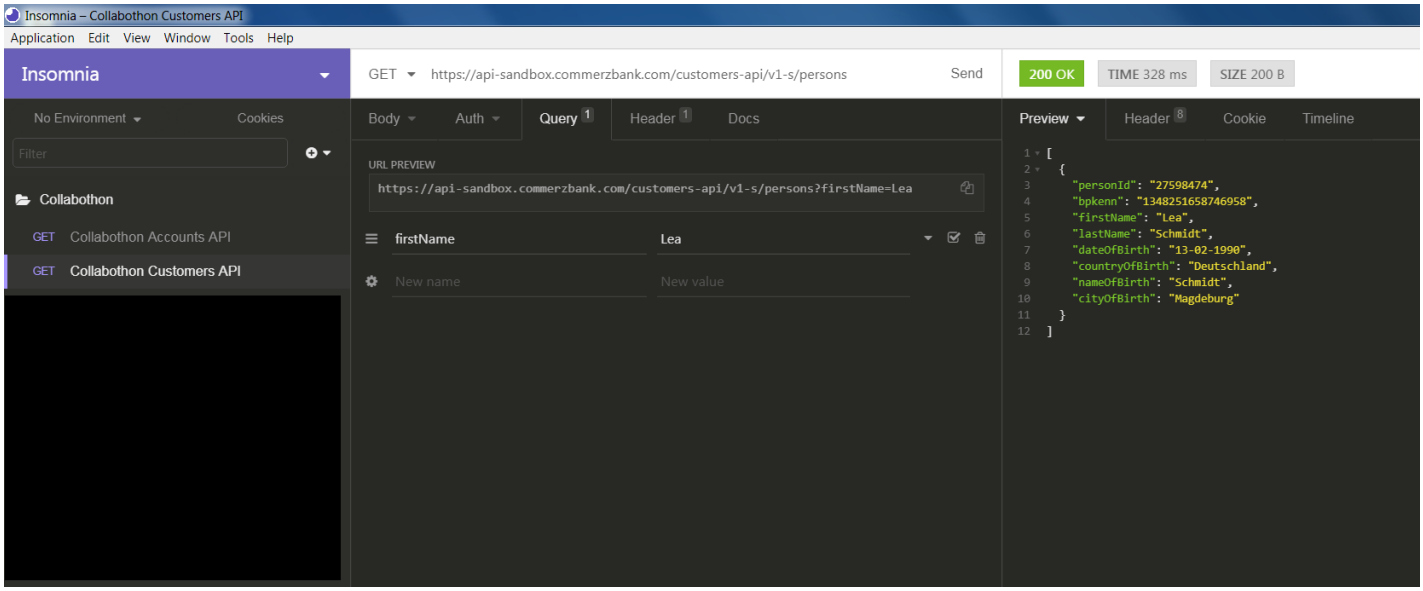
Using cURL to access the **Customers API**:



Using Insomnia to access the **Accounts API**:



Using Insomnia to access the **Customers API**:



Now that you have had your first contact with our APIs, you are ready to dive deeper into the APIs functionality. As outlined above, you can modify parts of the accounts informatic info about customers.

The Accounts API in detail

The Accounts API is used to receive information about accounts, such as balances, transactions and IBAN numbers. You have the following resources at hand:

GET	/accounts	Get a listing off all currently available accounts
Response:	HTTP 200 – OK	{ "accountList": [{

		<pre>"accountId": "string",
		"currency": "string",
		"iban": "string",
		"accountType": "string",
		"balances": [
		{
		"closingBooked": {
		"amount": {
		"currency": "string",
		"amount": 0
		},
		"date": "string",
		"lastActionDateTime": "string"
		}
		}
		]
		}
		]
		}</pre>

GET	/accounts/{accountId}	Get the details to an account specified by {accountId}

Response:	HTTP 200 OK	<pre>{
		"accountId": "string",
		"currency": "string",
		"iban": "string",
		"accountType": "string",
		"balances": [
		{
		"closingBooked": {
		"amount": {
		"currency": "string",
		"amount": 0
		},
		"date": "string",
		"lastActionDateTime": "string"
		}
		}
		]
		}</pre>

POST	/accounts/{accountId}	Add a new account – be sure to use the correct JSON model while posting

Response:	HTTP 201 – Created	<pre>{
		"accountId": "string",
		"currency": "string",
		"iban": "string",
		"accountType": "string",
		"balances": [
		{
		"closingBooked": {
		"amount": {
		"currency": "string",
		"amount": 0
		},
		"date": "string",
		"lastActionDateTime": "string"
		}
		}
		]
		}</pre>

DELETE	/accounts/{accountId}	Delete an account specified by {accountId}
Reponse:	HTTP 200 – OK	Deleted
GET	/accounts/{accountId}/balances	Get a listing of balances to an account specified by {accountId}

Response:	HTTP 200 – OK	<pre>{
		"closingBooked": {
		"amount": {
		"currency": "string",
		"amount": 0
		},
		"date": "string",
		"lastActionDateTime": "string"</pre>

		} }
GET	/accounts/{accountId}/transactions	Get a listing of transactions related to an account specified by {accountId}

Response:	HTTP 200 – OK	{
		"transactions": {
		"accountId": "string",
		"booked": [
		{
		"transactionId": "string",
		"bookingDate": "string",
		"valueDate": "string",
		"bookingTimestamp": "2018-08-21T11:15:50.078Z",
		"amount": {
		"currency": "string",
		"amount": 0
		}
		}
		],
		"pending": [
		{
		"transactionId": "string",
		"bookingDate": "string",
		"valueDate": "string",
		"bookingTimestamp": "2018-08-21T11:15:50.078Z",
		"amount": {
		"currency": "string",
		"amount": 0
		}
		}
		],
		"prospective": [
		{
		"transactionId": "string",
		"bookingDate": "string",
		"valueDate": "string",
		"bookingTimestamp": "2018-08-21T11:15:50.078Z",
		"amount": {
		"currency": "string",
		"amount": 0
		}
		}
		],
		"_links": {
		"viewAccount": "string"
		}
		}
		}

GET	/accounts/{accountId}/transactions/{transactionId}?bookingStatus={bookingStatus}	Get details to a transaction identified by {accountId} and {transactionId}. You need to specify your search with a naming the booking status of the transaction (booked, pending, prospective or both to see all). Example: */accounts/{accountId}/transactions?bookingStatus=both*

Response:	HTTP 200 – OK	{
		"transactionId": "string",
		"bookingDate": "string",
		"valueDate": "string",
		"bookingTimestamp": "string",
		"amount": {
		"currency": "string",
		"amount": 0
		}
		}

As you can see, you are also able to create a new account. When doing so, you send the information about the new account in the request body, formatted as JSON. If you send a n is unable to create an account and will respond with an Internal Server Error (500). You can use any account received through the GET /accounts method as template for your accou

The Customers API in detail

Use this API to search for customers stored in the database. As mentioned above, we came up with six individuals that all have their own accounts and transactions. You can look fo the Customers API and then use the Accounts API to get more detailed information on their bank accounts.

GET	/persons	Search for customers by search parameters. You can look for customers by first name (firstName), last name (lastName) date of bir city of birth ().

		Example: /customers?firstName=Sabine

Response:	HTTP 200 – OK	{
		"persons": [
		{
		"tenant": "string",
		"bpkenn": "string",
		"firstName": "string",
		"lastName": "string",
		"dateOfBirth": "string",
		"countryOfBirth": 0,
		"nameOfBirth": "string",
		"cityOfBirth": "string"
		}
		]
		}

GET	/persons/{personId}	Get details to a person identified by {personId}

Response:	HTTP 200 – OK	{
		"tenant": "string",
		"bpkenn": "string",
		"firstName": "string",
		"lastName": "string",
		"dateOfBirth": "string",
		"countryOfBirth": 0,
		"nameOfBirth": "string",
		"cityOfBirth": "string",
		"salutation": 0,
		"countryOfForeignTradeRegulations": 0,
		"countryOfTaxLiability": 0,
		"classificationOfEconomicActivities": 0,
		"titleOfNobility": "string",
		"salutationExtension": "string",
		"maritalStatus": 0,
		"individualSalutation": "string",
		"instituteNumber": "string",
		"employeeProtection": 0,
		"personellNumber": "string",
		"language": 0,
		"dateOfDeath": "string",
		"title": 0,
		"titleExtension": "string",
		"nationalities": [
		{
		"nationality": 0
		}
		],
		"industryCode": 0
		}

GET	/persons/{personId}/addresses	List the addresses to a person identified by {personId}

Response:	HTTP 200 – OK	{
		"addresses": [
		{
		"typeOfUsage": "string",
		"city": "string",
		"postalCountryCode": "string",
		"otherPartOfAddress": "string",
		"federalStateType": "string",
		"streetNumber": "string",
		"zipCode": "string",
		"street": "string"
		}
		]
		}

GET	/persons/{personId}/agreements	List the accounts associated with an a person identified by {personid}

Response:	HTTP 200 – OK	{
		"agreements": [
		{
		"accountId": "string"
		}
		]
		}

GET	/persons/{personId}/legitimation	Get the legitimation mode and information used by a person identified by {personId}

Response:	HTTP 200 – OK	{ "filingInCustomerId": "string", "dateOfExpiry": "string", "issuingAuthority": "string", "dateOfIssuance": "string", "placeOfIssuance": "string", "identificationType": "string", "identificationId": "string", "identificationCopyAvailable": "string", "
GET	/persons/{personId}/taxIds	List the tax IDs of a person identified by {personId}

Response:	HTTP 200 – OK	{
		"taxIds": [
		{
		"valueAddedTaxId": "string",
		"taxId": "string"
		}
		]
		}

Now you are ready to get hacking! Have fun!