convercus

User Management Api Implementation Manual v1.1.0

convercus

POINT 4MORE

Table of Content

1 User Management Api Implementation Manual

For all loyalty-related processes it is important to identify the person the system is interacting with. The central object in Convercus api for this identification is the **account** you can collect points on. This account can be linked to **user**-data, transactions, bookings, etc. Every account has a unique identifier, the accountId (e.g. 7d123457-bfa1-4a83-8213-123456789763), which is the technical ID all those connections are made with.
Additionally, every account can have multiple **identifiers** (i.e. card-codes, external identification-codes, etc.), which allow to make a connection to an account without the need to extract the account-id.
This documentation will explain the creation and maintanance of **accounts**, **users**, **identifiers** and **memberships** step-by-step and demonstrate, how all can be linked together to create a fully-fledged customer profile.

- https://staging.convercus.io/api-docs/swagger-ui.html (Staging Environment)
- https://api.convercus.io/api-docs/swagger-ui.html (Production Environment)

and on the Developer Page

- https://developer.convercus.io/

2 Change Log

Version	Change Date	Change Log
v1.1.0	2021-03-12	• Minor corrections
v1.0.0	2020-04-30	• Initial document

3 General Information

3.1 Authentification

Every request requires a JWT-Token for authentication. The token can be obtained with the following request:

```
curl --location --request POST '{{api_url}}/auth/login' \
--header 'Content-Type: application/json' \
--data-raw '{
  "org": "{{org}}",
  "userName": "{{userName}}",
  "password": "{{password}}"
}'
```

where the following variables have been used:

Variable	Description
{{api_url}}	Endpoint of the api. • https://staging.convercus.io (Staging) • https://api.convercus.io (Production)
{{org}}	Organization code for correct mapping. This value will be given by Convercus.
{{userName}}	User-Name of the api-user. This value will be given by Convercus.
{{password}}	User-Password of the api-user. This value will be given by Convercus.

The JWT-Token can be found in the body of the response.

Note, that the token expires after 24 hours. To have a valid token at all times, it is necessary to generate the token on a regular basis.

3.2 Loyalty Elements Overview

Note, that due to structure of the platform, there are essentially three objects which contain information about the user, his account and membership. Depending on the set of information you are interested in, you may need to create and maintain all of them.

- **account:**
  - This is the central element of the loyalty system.
  - Every account has its unique accountId. All relevant loyalty-processes can be linked to this id.
  - N identifiers of different id-type may serve as additional external identifiers for an account. Usually, these identifiers are Cardcodes or external numbers (like a online-shop-id).
  - Accounts can be anonymous, if they don't have user-information connected (see membership).
- **user:**
  - This is the object which contains personal user-data like name, address, etc.
  - User data may be created independently from an account. Without the connection to an account (see membership) however, there is no way to interact with this data in a loyalty context (e.g. you cannot earn points on a user, but an account.)
- **membership:**
  - This object connects an account to a user object.
  - The creation of a membership is typically the result of a completed registration.
  - Optins for the program are related to this object.

A straight-forward implementation of this will be explained in the following.

4 Accounts

4.1 Creating Accounts

To earn points with the loyalty system, the customer needs an **account**. This account can later be linked (with a membership) to a user profile that contains user data. However, this is optional and loyalty points can be obtained and spent with an (anoymous) account by itself.

To create an account, use:

```
curl --location --request POST '{{api_url}}/accounts' \
--header 'interaction-id: {{interactionId}}' \
--header 'Authorization: {{jwt_token}}' \
--header 'Content-Type: application/json' \
```

with

Variable	Description
{{api_url}}	Endpoint of the api. • https://staging.convercus.io (Staging) • https://api.convercus.io (Production)
{{jwt_token}}	The JWT-token, which has been generated by authentication.
{{interactionId}}	Unique Identifier of the cash machine (or virtual equivalent, e.g. online-shop), which produced the receipt. This ID has to be listed in the Convercus System as with this ID, the connection to the call origin is made.

The account will be created in the database and is assigned an accountId, which can be obtained from the response's header section Location:

Location	{{api_url}}/accounts/accountId

4.2 Getting Account Details

As stated before, the account is the center of the whole loyalty system. The accountId can be used as common connection ID for practically all loyalty connections.

You can get the basic account information with the following request:

```
curl --location --request GET '{{api_url}}/accounts/{{accountId}}' \
--header 'Authorization: {{jwt_token}}' \
--header 'interaction-id: {{interactionId}}' \
--header 'id-type: {{idType}}' \
--header 'Content-Type: application/json' \
```

with variables

Variable	Description
{{api_url}}	Endpoint of the api. • https://staging.convercus.io (Staging) • https://api.convercus.io (Production)
{{accountId}}	ID of the account that needs to be checked. The ID has to be given in the format, that is dictated by the {{idType}},
{{jwt_token}}	The JWT-token, which has been generated by authentication.
{{interactionId}}	Unique Identifier of the cash machine (or virtual equivalent, e.g. online-shop), which produced the receipt. This ID has to be listed in the Convercus System as with this ID, the connection to the call origin is made.
{{idType}}	Identifier-Type (corresponding to the {{accountId}}). **Available values:** • APPCODE (e.g. A1B2C3D4E5) • CARDCODE (e.g. V1W2X3Y4Z5) • EXTERNALCODE (e.g. 123456780123) • ID (account-identifier, e.g. 7d123457-bfa1-4a83-8213-123456789763)

Note, that it's possible to get the account-object using the identifier code (e.g. {{idType}}=CARDCODE, {{accountId}}=V1W2X3Y4Z5) or the accountId itself (e.g. {{idType}}=ID, {{accountId}}=7d123457-bfa1-4a83-8213-123456789763). Thus, the request may also be used to extract an accountId from a given Identifier.

The account-object per se is rather slender, only containing an ID (accountId), a program reference and a status of the account.

```
{
 "id": "550e8400-e29b-11d4-a716-446655440000",
 "program": "Pgr-A",
 "status": "ACTIVE"
}
```

If your program allows deactivation or locking of accounts, you should make sure, that accounts which don't have the status ACTIVE cannot proceed with the following earn- and burn-processes. Deleted Accounts will not be responded at all.

4.3 Getting Account Balance

The account's balance can be checked with the following command:

```
curl --location --request GET '{{api_url}}/accounts/{{accountId}}/balance' \
--header 'Authorization: {{jwt_token}}' \
--header 'interaction-id: {{interactionId}}' \
--header 'id-type: {{idType}}' \
--header 'Content-Type: application/json' \
```

with variables

Variable	Description
{{api_url}}	Endpoint of the api. • https://staging.convercus.io (Staging) • https://api.convercus.io (Production)
{{accountId}}	ID of the account that needs to be checked. The ID has to be given in the format, that is dictated by the {{idType}}.
{{jwt_token}}	The JWT-token, which has been generated by authentication.
{{interactionId}}	Unique Identifier of the cash machine (or virtual equivalent, e.g. online-shop), which produced the receipt. This ID has to be listed in the Convercus System as with this ID, the connection to the call origin is made.
{{idType}}	Identifier-Type (corresponding to the {{accountId}}). **Available values:** • APPCODE (e.g. A1B2C3D4E5) • CARDCODE (e.g. V1W2X3Y4Z5) • EXTERNALCODE (e.g. 123456780123) • ID (account-identifier, e.g. 7d123457-bfa1-4a83-8213-123456789763)

This will return the value of points available to the account. Loyalty points can be locked, due to refund periods or individual reasons:

```
{
  "points": 100,
  "lockedPoints": 0
}
```

5 Identifiers

An identifier can be used to simplify the access to a customers account to gather loyalty points. By linking the identifier (that can be a loyalty customer card, online identifier or others) to a customer's accountId, they can verify at the cash register or online checkout to be the customer represented by said accountId.
When an identifier is linked to an account, it can be used as additional identification of the account (can be used as {{accountId}} variable in calls but with different idType).

5.1 Creating Identifiers

A new identifier can be created and mapped to the account.

```
curl --location --request POST '{{api_url}}/accounts/{{accountId}}/identifiers' \
--header 'interaction-id: {{interactionId}}' \
--header 'Authorization: {{jwt_token}}' \
--header 'Content-Type: application/json' \
--data-raw '{{body}}'
```

with

Variable	Description
{{api_url}}	Endpoint of the api. • https://staging.convercus.io (Staging) • https://api.convercus.io (Production)
{{accountId}}	Unique ID assigned to an account at creation.
{{jwt_token}}	The JWT-token, which has been generated by authentication.
{{interactionId}}	Unique Identifier of the cash machine (or virtual equivalent, e.g. online-shop), which produced the receipt. This ID has to be listed in the Convercus System as with this ID, the connection to the call origin is made.
{{body}}	Body with Identifier-Information as explained in the following.

The body may look like this:

```
{
  "code": "ABCD0001",
  "displayCode": "ABCD0001",
  "type": "APPCODE"
}
```

with

Attribute	Description	Relevance
code	Code that can be used to identify an account	Optional
displayCode	Display value of the identifier e.g. for printing	Optional
type	Declares the type of identifier and has to be one of: • APPCODE • CARDCODE • EXTERNALCODE	**Mandatory**

5.2 Getting Identifiers

All identifiers assigned to given account can be retrieved simultaniously:

```
curl --location --request GET '{{api_url}}/accounts/{{accountId}}/identifiers' \
--header 'interaction-id: {{interactionId}}' \
--header 'Authorization: {{jwt_token}}' \
--header 'id-type: {{idType}}' \
```

with

Variable	Description
{{api_url}}	Endpoint of the api. • https://staging.convercus.io (Staging) • https://api.convercus.io (Production)
{{accountId}}	Unique ID assigned to an account at creation. The ID has to be given in the format, that is dictated by the {{idType}}.
{{jwt_token}}	The JWT-token, which has been generated by authentication.
{{interactionId}}	Unique Identifier of the cash machine (or virtual equivalent, e.g. online-shop), which produced the receipt. This ID has to be listed in the Convercus System as with this ID, the connection to the call origin is made.

{{idType}}	Identifier-Type (corresponding to the {{accountId}}). **Available values:** • APPCODE (e.g. A1B2C3D4E5) • CARDCODE (e.g. V1W2X3Y4Z5) • EXTERNALCODE (e.g. 123456780123) • ID (account-identifier, e.g. 7d123457-bfa1-4a83-8213-123456789763)

This will return all identifiers with their specific identifier type the account is currently assigned to, e.g.:


```
[
  {
    "identifierId": "7b7587da-7467-4fe6-96d5-1d650a49f1fc",
    "code": "TESTCODE",
    "displayCode": "AB1299CD",
    "type": "APPCODE",
    "status": "ACTIVE"
  }
]
```


In order to secure a constistent mapping when displaying the codes, it is hightly recommened to set up different codes and code types unambigously.

6 Users

6.1 Creating a new User

A new user can be created with:

```
curl --location --request POST '{{api_url}}/users' \
--header 'Authorization: {{jwt_token}}' \
--header 'interaction-id: {{interactionId}}' \
--header 'Content-Type: application/json' \
--data-raw '{{body}}'
```

where

Variable	Description
{{api_url}}	Endpoint of the api. • https://staging.convercus.io (Staging) • https://api.convercus.io (Production)
{{jwt_token}}	The JWT-token, which has been generated by authentication.
{{interactionId}}	Unique Identifier of the cash machine (or virtual equivalent, e.g. online-shop), which produced the receipt. This ID has to be listed in the Convercus System as with this ID, the connection to the call origin is made.
{{body}}	Body with user data parameters, specified in the following.

The body may look like this:

```
{
   "emailAddress": "test-mail@mail.com",
   "givenName": "testName",
   "familyName": "testFamilyname",
   "streetHouseNo": "testStreet",
   "zipCode": "12345",
   "city": "testCity",
   "countryCode": "DE",
   "genderCode": "FEMALE",
   "birthDate": "1991-01-01",
   "phone": "012345",
   "customProperties": [
      {
         "name": "exampleCustomProperty 1",
         "value": "abcde"
      },
      {
```

```
        "name": "exampleCustomProperty 1",
        "value": "10001"
      }
   ]
}
```

with (all string-type)

Attribute	Description	Relevance
emailAddress	E-Mail of customer	**Mandatory**
givenName	First name of customer	Optional
familyName	Family name of customer	Optional
streetHouseNo	Adress of customer	Optional
zipCode	Zip-Code of customer	Optional
city	City of residence	Optional
countryCode	Country of residence, with code specified in “ISO 3166-1 alpha-2”	Optional
genderCode	Gender, can be one of: • MALE • FEMALE • DIVERSE	Optional
birthDate	Date of birth in format YYYY-MM-DD: e.g. "1991-01-01"	Optional
phone	Phone number of customer	Optional
customProperties	array with additional properties in pairs of name: string, value: string, that have been preconfigured.	Optional

The user will be created in the database and is assigned a userId, which can be obtained from the response’s header section Location:

Location	{{api_url}}/users/userId

6.2 Displaying User properties

The user's properties can be retrieved with the unique userId a user is assigned with at creation. This userId may also be extracted from a membership.

```
curl --location --request GET '{{api_url}}/users/{{userId}}/' \
--header 'Authorization: {{jwt_token}}' \
--header 'interaction-id: {{interactionId}}' \
--header 'Content-Type: application/json' \
```

with

Variable	Description
{{api_url}}	Endpoint of the api. • https://staging.convercus.io (Staging) • https://api.convercus.io (Production)
{{userId}}	The unique identification a user was assigned at creation. Alternatively, this is taken from a membership-connection.
{{jwt_token}}	The JWT-token, which has been generated by authentication.
{{interactionId}}	Unique Identifier of the cash machine (or virtual equivalent, e.g. online-shop), which produced the receipt. This ID has to be listed in the Convercus System as with this ID, the connection to the call origin is made.

This will retrieve all properties the user has, e.g.


```
{
  "userId": "dd820ab2-6833-4ee3-9add-7bebf4dadddd",
  "emailAddress": "test-mail@mail.com",
  "givenName": "testName",
  "familyName": "testFamilyname",
  "streetHouseNo": "testStreet",
  "zipCode": "12345",
  "city": "testCity",
  "countryCode": "DE",
  "genderCode": "FEMALE",
  "birthDate": "1991-01-01",
  "phone": "012345",
  "customProperties": [
    {
      "name": "exampleCustomProperty 1",
      "value": "abcde"
    }
  ]
}
```

again with

Attribute	Description	Relevance
emailAddress	E-Mail of customer	**Mandatory**
givenName	First name of customer	Optional
familyName	Family name of customer	Optional
streetHouseNo	Address of customer	Optional
zipCode	Zip-Code of customer	Optional
city	City of residence	Optional
countryCode	Country of residence, with code specified in “ISO 3166-1 alpha-2”	Optional
genderCode	Gender, can be one of: • MALE • FEMALE • DIVERSE	Optional
birthDate	Date of birth in format YYYY-MM-DD: e.g. "1991-01-01"	Optional
phone	Phone number of customer	Optional
customProperties	array with additional properties in pairs of name: string, value: string, that have been preconfigured.	Optional

Note, that the content of this response may differ with the program. There may be multiple customProperties, which are completely program-specific. Furthermore, it is possible that in future versions, the response will be expanded by more fields, so you should make sure to be able to accept more output.

6.3 Updating User Information

Using a users unique userId, a user can be updated in two variants:

6.3.1 OVERWRITE ALL USER DATA (PUT)

A **PUT** request will update the user data given in the argument, **deleting all pre-existing properties** that were not specified in the request.

```
curl --location --request PUT '{{api_url}}/users/{{userId}}' \
--header 'Authorization: {{jwt_token}}' \
--header 'interaction-id: {{interactionId}}' \
--header 'Content-Type: application/json' \
--data-raw '{{body}}'
```

with

Variable	Description
{{api_url}}	Endpoint of the api. • https://staging.convercus.io (Staging) • https://api.convercus.io (Production)
{{userId}}	The unique identification a user was assigned at creation.
{{jwt_token}}	The JWT-token, which has been generated by authentication.
{{interactionId}}	Unique Identifier of the cash machine (or virtual equivalent, e.g. online-shop), which produced the receipt. This ID has to be listed in the Convercus System as with this ID, the connection to the call origin is made.
{{body}}	Body with user data parameters, specified in the following.

The body may look like this:

```
{
   "emailAddress": "test2-mail@mail.com",
   "givenName": "testName",
   "familyName": "testFamilyname",
   "streetHouseNo": "testStreet",
   "zipCode": "22222",
   "city": "testCity",
   "countryCode": "DE",
   "genderCode": "FEMALE",
   "birthDate": "1991-01-01",
   "phone": "999999999",
   "customProperties": [
      {
         "name": "exampleCustomProperty 1",
         "value": "abcde"
      }
   ]
}
```

with

Attribute	Description	Relevance
emailAddress	E-Mail of customer	**Mandatory**
givenName	First name of customer	Optional
familyName	Family name of customer	Optional

streetHouseNo	Address of customer	Optional
zipCode	Zip-Code of customer	Optional
city	City of residence	Optional
countryCode	Country of residence, with code specified in “ISO 3166-1 alpha-2”	Optional
genderCode	Gender, can be one of: • MALE • FEMALE • DIVERSE	Optional
birthDate	Date of birth in format YYYY-MM-DD: e.g. "1991-01-01"	Optional
phone	Phone number of customer	Optional
customProperties	array with additional properties in pairs of name: string, value: string, that have been preconfigured.	Optional

As mentioned before, this will **delete all properties**, that are **not specified** in the request.

6.3.2 OVERWRITE SELECTED USER DATA (PATCH)

Contrary, a **PATCH** request will change all properties given in the argument, while **leaving all arguments** not specified in the request **unmodified**.

```
curl --location --request PATCH '{{api_url}}/users/{{userId}}' \
--header 'Authorization: {{jwt_token}}' \
--header 'interaction-id: {{interactionId}}' \
--header 'Content-Type: application/json' \
--data-raw '{{body}}'
```

with

Variable	Description
{{api_url}}	Endpoint of the api. • https://staging.convercus.io (Staging) • https://api.convercus.io (Production)
{{userId}}	The unique identification a user was assigned at creation.
{{jwt_token}}	The JWT-token, which has been generated by authentication.

{{interactionId}}	Unique Identifier of the cash machine (or virtual equivalent, e.g. online-shop), which produced the receipt. This ID has to be listed in the Convercus System as with this ID, the connection to the call origin is made.
{{body}}	Body with user data parameters, specified in the following.

The body with parameters may look like this:

```
{
  "zipCode": 22222
}
```

with the same possible parameters as the PUT request:

Attribute	Description	Relevance
emailAddress	E-Mail of customer	Optional
givenName	First name of customer	Optional
familyName	Family name of customer	Optional
streetHouseNo	Address of customer	Optional
zipCode	Zip-Code of customer	Optional
city	City of residence	Optional
countryCode	Country of residence, with code specified in “ISO 3166-1 alpha-2”	Optional
genderCode	Gender, can be one of: • MALE • FEMALE • DIVERSE	Optional
birthDate	Date of birth in format YYYY-MM-DD: e.g. "1991-01-01"	Optional
phone	Phone number of customer	Optional
customProperties	array with additional properties in pairs of name: string, value: string, that have been preconfigured. **Note**: customProperties always requires **all** properties, even if only one or a few need to be changed. All properties that are not specified, even if they remain unchanged, will be deleted.	Optional

The examplary request will update the specified properties in the data-raw argument, in this case the zipCode.
Individual properties can be deleted by updating them with null, e.g.:

```
curl --location --request PATCH '{{api_url}}/users/{{userId}}' \
--header 'Authorization: {{jwt_token}}' \
--header 'interaction-id: {{interactionId}}' \
--header 'Content-Type: application/json' \
--data-raw '{
  "zipCode": null
}'
```

Note: When updating the custom properties array, each entry has to be sent with the request, as both PUT and PATCH can only update the array in its entirety.

6.4 Deleting Users

A user can be deleted from the database, by sending a DELETE request with the unique userId.

```
curl --location --request DELETE '{{api_url}}/users/{{userId}}' \
--header 'Authorization: {{jwt_token}}' \
--header 'interaction-id: {{interactionId}}' \
--header 'Content-Type: application/json' \
```

again with

Variable	Description
{{api_url}}	Endpoint of the api. • https://staging.convercus.io (Staging) • https://api.convercus.io (Production)
{{userId}}	The unique identification a user was assigned at creation.
{{jwt_token}}	The JWT-token, which has been generated by authentication.
{{interactionId}}	Unique Identifier of the cash machine (or virtual equivalent, e.g. online-shop), which produced the receipt. This ID has to be listed in the Convercus System as with this ID, the connection to the call origin is made.

This will delete the user and anonymize the user data while simultaniously resetting the loyalty points to 0.

7 Memberships

A user, which is a customer profile with customer-specific data, is not able to collect loyalty points by itself, since the loyalty system collects points on accounts only. A membership can link a customer profile to an account, enabling the user to collect points for his connected accountId. One account may have N user connected with N memberships (always connect 1 user per membership). In practise, the most common situation is the connection 1 account : 1 membership : 1 user on which we will mainly focus in the following.
Note that the membership is the object which contains optin-information as optins are typically set / activated when a registration is completed (= user connected with account).

7.1 Creating Memberships

A new membership for given accountId can be created to connect a **user** with an **account**.

```
curl --location --request POST '{{api_url}}/accounts/{{accountId}}/memberships' \
--header 'Authorization: {{jwt_token}}' \
--header 'interaction-id: {{interactionId}}' \
--header 'Content-Type: application/json' \
--data-raw '{{body}}'
```

with variables

Variable	Description
{{api_url}}	Endpoint of the api. • https://staging.convercus.io (Staging) • https://api.convercus.io (Production)
{{accountId}}	ID of the account that will get the membership connection.
{{jwt_token}}	The JWT-token, which has been generated by authentication.
{{interactionId}}	Unique Identifier of the cash machine (or virtual equivalent, e.g. online-shop), which produced the receipt. This ID has to be listed in the Convercus System as with this ID, the connection to the call origin is made.
{{body}}	Body with membership data parameters, specified in the following.

The body may look like this:


```
{
 "accountId": "ff2bd33f-3878-4cfc-9ecc-1a541972e498",
 "memberRole": "OWNER",
 "optins": [
  {
   "flag": true,
   "type": "Post"
  },
  {
   "flag": true,
   "type": "Newsletter"
  }
 ],
 "partnerId": "550e8400-e29b-11d4-a716-446655440000",
 "userId": "d22917e4-e764-4f12-8969-b834673f3acd"
}
```


with

Attribute	Description	Relevance
accountId	ID of the account that will receive the transaction.	**Mandatory**
memberRole	Indicates the relation the user has to the account. Must be one of: • OWNER • COOWNER • COLLECTOR Note that the first membership has to have the OWNER-role. There can be only one OWNER per membership. If not explicitly required otherwise, only one user should be connected via one membership to one account.	**Mandatory**
optins	array with pairs of flag: boolean, type: string with set optins (type) and there value (flag).	Optional
partnerId	Declares the Id for the loyalty partner the membership should be linked to. This can be used to grant bonuses, like welcome-bonus or seasonal bonuses.	Optional / **Mandatory for multipartner programs**

The response will be similar to previous Creation-Requests, where the crucial information can be obtained from the Location row of the response header array.
The composition is:

Location	{{api_url}}/accounts/{{accountId}}/memberships/{{membershipId}}

Where {{accountId}} and {{membershipId}} are the values of given account and account-membership, respectively.

7.2 Getting Memberships of an Account

All memerships (and optins) of given account can be retrieved with the account identification:

```
curl --location --request GET '{{api_url}}/accounts/{{accountId}}/memberships' \
--header 'Authorization: {{jwt_token}}' \
--header 'interaction-id: {{interactionId}}' \
--header 'id-type: {{idType}}' \
--header 'Content-Type: application/json' \
```

with variables

Variable	Description
{{api_url}}	Endpoint of the api. • https://staging.convercus.io (Staging) • https://api.convercus.io (Production)
{{accountId}}	ID of the account that should be checked.. The ID has to be given in the format, that is dictated by the{{idType}},
{{jwt_token}}	The JWT-token, which has been generated by authentication.
{{interactionId}}	Unique Identifier of the cash machine (or virtual equivalent, e.g. online-shop), which produced the receipt. This ID has to be listed in the Convercus System as with this ID, the connection to the call origin is made.
{{idType}}	Identifier-Type (corresponding to the {{accountId}}). **Available values:** • APPCODE (e.g. A1B2C3D4E5) • CARDCODE (e.g. V1W2X3Y4Z5) • EXTERNALCODE (e.g. 123456780123) • ID (account-identifier, e.g. 7d123457-bfa1-4a83-8213-123456789763)

Note again, that you can use the identifier code directly (e.g. {{idType}}=CARDCODE, {{accountId}}=V1W2X3Y4Z5) at this point.

The returned object looks like this:


```
[
 {
  "accountId": "550e8400-e29b-11d4-a716-446655440000",
  "memberRole": "OWNER",
  "membershipId": "550e8400-e29b-11d4-a716-446655440000",
  "optins": [
   {
    "flag": true,
    "type": "email"
   }
  ],
  "partnerId": "550e8400-e29b-11d4-a716-446655440000",
  "userId": "550e8400-e29b-11d4-a716-446655440000"
 }
]
```


where

Attribute	Description
accountId	Technical account-Id connected to this membership.
memberRole	Role of the membership. In general, it is possible to have multiple memberships, one account owner ("memberRole": "OWNER") and several COOWNERs or COLLECTORs. In the standard setup however, there is only one membership (with "memberRole": "OWNER"), connecting one account to one user.
membershipId	Technical ID of the membership.
optins	List of optins that this membership has. Optin names are denoted by type, optin values are denoted by flag.
partnerId	Technical ID of the partner, the membership is belonging to.
userId	Technical ID of the user connected to an account via the membership.

If no membership is returned, the returned array is empty []. Depending on the specific program setup, these users may not be allowed to do anything if there are not registered. This logic has to be adopted here if that's the case.

7.3 Getting a specific Membership of an Account

To investigate a specific membership of given accountId closer, all enlisted information can be obtained at once by using:

```
curl --location --request GET '{{api_url}}/accounts/{{accountId}}/memberships/{{membershipId}}' \
--header 'Authorization: {{jwt_token}}' \
--header 'interaction-id: {{interactionId}}' \
--header 'id-type: {{idType}}' \
--header 'Content-Type: application/json' \
```

with

Variable	Description
{{api_url}}	Endpoint of the api. • https://staging.convercus.io (Staging) • https://api.convercus.io (Production)
{{accountId}}	ID of the account that should be checked.. The ID has to be given in the format, that is dictated by the{{idType}},
{{membershipId}}	Unique ID of the membership.
{{jwt_token}}	The JWT-token, which has been generated by authentication.
{{interactionId}}	Unique Identifier of the cash machine (or virtual equivalent, e.g. online-shop), which produced the receipt. This ID has to be listed in the Convercus System as with this ID, the connection to the call origin is made.
{{idType}}	Identifier-Type (corresponding to the {{accountId}}). **Available values:** • APPCODE (e.g. A1B2C3D4E5) • CARDCODE (e.g. V1W2X3Y4Z5) • EXTERNALCODE (e.g. 123456780123) • ID (account-identifier, e.g. 7d123457-bfa1-4a83-8213-123456789763)

The response object looks like this:


```
{
  "membershipId": "4670c014-45b4-4d3b-bafd-b4bd13b5d659",
  "accountId": "fd313081-11ae-4e50-b429-5c1cc0bfb357",
  "userId": "24632823-71ca-47d3-8e20-6f9c7d727ef5",
  "memberRole": "OWNER",
  "partnerId": null,
  "terminated": null,
  "createdAt": null,
  "optins": [
   {
    "flag": true,
    "type": "email"
   }
  ]
}
```


where

Attribute	Description
membershipId	Technical ID of the membership.
accountId	Technical account-Id connected to this membership.
userId	Technical ID of the user connected to an account via the membership.
memberRole	Role of the membership. In general, it is possible to have multiple memberships, one account owner ("memberRole": "OWNER") and several COOWNERs or COLLECTORs. In the standard setup however, there is only one membership (with "memberRole": "OWNER"), connecting one account to one user.
partnerId	Technical ID of the partner, the membership is belonging to.
optins	List of optins that this membership has. Optin names are denoted by type, optin values are denoted by flag.

7.4 Update Membership Optins

The optins for a specific membership of given accountId can be updated by reassining them in the following request:

```
curl --location --request PATCH '{{api_url}}/accounts/{{accountId}}/memberships/{{membershipId}}' \
--header 'Authorization: {{jwt_token}}' \
--header 'interaction-id: {{interactionId}}' \
--header 'id-type: {{idType}}' \
--header 'Content-Type: application/json' \
--data-raw '{{body}}'
```

with

Variable	Description
{{api_url}}	Endpoint of the api. • https://staging.convercus.io (Staging) • https://api.convercus.io (Production)
{{accountId}}	ID of the account that should be checked.. The ID has to be given in the format, that is dictated by the{{idType}},
{{membershipId}}	Unique ID of the membership.
{{jwt_token}}	The JWT-token, which has been generated by authentication.
{{interactionId}}	Unique Identifier of the cash machine (or virtual equivalent, e.g. online-shop), which produced the receipt. This ID has to be listed in the Convercus System as with this ID, the connection to the call origin is made.
{{idType}}	Identifier-Type (corresponding to the {{accountId}}). **Available values:** • APPCODE (e.g. A1B2C3D4E5) • CARDCODE (e.g. V1W2X3Y4Z5) • EXTERNALCODE (e.g. 123456780123) • ID (account-identifier, e.g. 7d123457-bfa1-4a83-8213-123456789763)
{{body}}	Body with Identifier-Information as explained in the following.

The body may look like this:

```
[
    {
      "flag": true,
      "type": "email"
    }
]
```

where each updated optin must be paired with a true-false-flag.
Note: body musst contain **all** optins, even if they remain unchanged. Any optin not specified in body will be **deleted**.