CTC Wallet – API description for carriers

Version 1.2

Trusted Carrier GmbH & Co. KG
Breitenbachstraße 1
D-60487 Frankfurt
Telephone: +49 (0) 89 890 569 – 280
Internet: www.trusted-carrier.com

Versions

Version	Date	Type of change
1.2	08.07.24	Added chapter 3 (MFA login/logout events)
1.1	30.04.24	Updates to Security mode
1.0	06.03.24	First official version

Table of contents

1 General information 4
1.1 CTC Staging system 4
1.2 Structure of requests 4
1.3 Structure of answers 5
1.4 Notifications to external systems & security mode 5
2 Methods to manage driver masterdata 7
2.1 Get information about a single driver 7
2.2 Get information about multiple drivers 9
2.3 Create driver 10
2.4 Invite driver 12
2.5 Confirm driver request 15
2.6 Delete request 16
2.7 Remove driver 17
2.8 Change password 18
2.9 Remove phone number for SMS-TAN process 19
2.10 Update driver information 20
3 Methods to manage driver authentication in Multi-User installations 21
3.1 MFA Login event 22
3.2 MFA Logout event 23
4 Notifications to external systems 25
4.1 Driver accepts association request by the carrier 25
4.2 Driver rejects association request by the carrier 26
4.3 Driver removes an existing association 27
4.4 Driver requests an association 28
4.5 Driver deletes certificate issued by the carrier 29

1 General information

The Wallet Asset API is exposing all functions to users that are also available in the CTC web frontend.

1.1 CTC Staging system

To test implementing the CTC Wallet API, we recommend using the CTC Staging system, which is fully separated from the Production system. Please contact us if you are interested in an account.

The Staging system uses different URIs compared with the Production System.

Example:

Staging: https://walletapi-staging.trusted-carrier.com/api/restAPIs/v1/carrier/getDrivers

Production: https://walletapi.trusted-carrier.com/api/restAPIs/v1/carrier/getDrivers

From here on, this document only names the Production URIs (except for Staging-only methods mentioned in chapter 4).

1.2 Structure of requests

All requests are POST. The header must contain the information about the content type.

If the security mode (see also chapter 1.4) is “Standard”, the authorization happens using the CTC-generated public key in the field “publicKey” within the body.

If the security mode is “JWT”, the token must be part of the “authorization” field within the header and must include your company ID.

Example header (with “JWT”):

```
{
    "Content-Type": "application/json;charset=UTF-8",
    "Authorization": "Bearer <JSON Web Token>"
}
```

Example body (with “Standard”):

```
{
    "publicKey": "<ctc-generated-publickey>",
    "data": {
```

```
        <data>
    }
}
```

Your keys are available on the page “API settings” in the “Admin” module.

1.3 Structure of answers

The answer starts with an error code. If the request was successful, the “code” field is 0. There may be a “warnings” field, if, e.g., some part of the request is no longer supported and only kept for compatibility reasons.

There may be an additional “payload” field, containing details about the vehicle component.

```
{
    "error_code": {
        "code": "0",
        "message": "Success",
        "warnings": "ABC is deprecated"
    },
    "payload": {
        <payload>
    }
}
```

If there are any errors, the “error_code” field contains a “message” and a “details” field which describes the error.

Example:

```
{
    "error_code": {
        "code": "4001",
        "message": "Invalid parameters",
        "details": "Driver not found"
    }
}
```

If the request is not successful, no changes to a driver association are done.

1.4 Notifications to external systems & security mode

When a support process for a vehicle component is finished, the corresponding information is sent to your system, if an endpoint has been defined on the screen “API settings” in the “Admin module” of your CTC account (see chapter 5).

You can choose the mode of security on the “API settings” screen:

- “Standard”: This is the basic security setting, allowing authorization with a pre-shared key sent in the body of the request.
    - URL: In this field, the endpoint of the external system can be specified. If this field has a value, the ERP key field must also be filled
    - ERP key: CTC will send the value entered here towards the external system, when communication is initiated by CTC
    - CTC key: CTC creates a key, that the external system needs to send in all requests when communication is initiated by the external system
- “JWT”: This extends the security with JSON Web Tokens (JWT) in the authentication header, which must be sent by both CTC and the external system in each request.
    - URL: In this field, the endpoint of the external system can be specified. If this field has a value, the ERP JWT key field must also be filled
    - CTC JWT key: CTC will create a private/public keypair in RSA2048 format
    - ERP JWT key: The value entered here must be in RSA2048 & ASN.1 format
      Each location will be have security key generated by CTC

If JWT is enabled:

- Both carrier and CTC APIs will be required to send an authorization header with the value: Bearer <JWT token>
- JWT settings
    - RS512
    - Contents of the payload
        - iss: sender URL
            - CTC will use one of the following: https://ctc.trusted-carrier.com/ (production system), https://ctc-staging.trusted-carrier.com/ (staging system)
            - While “iss” is not a mandatory value, the external system is advised to provide it and it will be stored in the CTC logs
        - cid: your company ID
        - iat: unixtime seconds

- Timestamp when the token was issued
- Must be before current time

■ exp: unixtime seconds

- Timestamp when the token expires
- Maximum of 60 seconds into the future

- Example token sent by an external system:
  - eyJhbGciOiJSUzUxMiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3MjAwMDc5NjYsImlhdCI6MTcyMDAwNzkwNiwiaXNzIjoieW91cmNvbXBhbnkuY29tIiwiY2lkIjoxMjN9.HZSfVBOIvDSS3vRnTF-4CaV6OF4GEZF5ETIRwhc9qihFrHfV7nnv1V25n3laiTXtsdvqJP5GKuWkflummqOqg7WxU7Du-9WUOdyeXQAt-gIlS6nl7ojKGde7TqZ6JvIC-OE-f6guVkw2mWJkycSFe_g7QBEeQ9bXa3SIzW9dydQEQx_fo8W3DLJau5bn-vANhVLP5HEo-6YEd0aomph7Oxvtrt9h5BiOONtUThy-jpnBk_kRO9vyblOwcOQYk0_plHHGRTNEhGGOxIjhokU5p9qlsc-P1TiPQlC8FgleW3ydwOxJt5_-g_YrqzrhcOEdIss_DzDkYP7_PQWXNZTidg
  - Decoded red part (header):

    ```
    {
      "alg": "RS512",
      "typ": "JWT"
    }
    ```

  - Decoded green part (payload):

    ```
    {
      "exp": 1720007966,
      "iat": 1720007906,
      "iss": "yourcompany.com",
      "cid": 123
    }
    ```

  - Blue part is to verify the signature
- The request will be rejected with a 401 response if:
  - Authorization header is missing or malformed, and the account has JWT enabled
  - JWT isn't verifiable using the public key provided by location
  - JWT is expired

2 Methods to manage driver masterdata

2.1 Get information about a single driver

URI: api/restAPIs/v1/carrier/getDriver

2.1.1 Functional description

This method returns current available information about a single driver.

Data:

- *userID*: (mandatory)
  User ID of the driver account. The ID must belong to either a personal driver account with a completed signup process, or a corporate driver created by the requesting carrier.

2.1.2 Example data exchange

Request sent to CTC Wallet:


```
{
    "publicKey": "ctc-generated-publickey",
    "data": {
        "userID": 1128
    }
}
```


Response from CTC Wallet:


```
{
    "error_code": {
        "code": "0",
        "message": "Success"
    },
    "payload": {
        "userID": 1128,
        "firstName": "Max",
        "lastName": "Mustermann",
        "nationality": "DE",
        "primaryLanguage": "de",
        "callingCode": "49",
        "telephoneNumber": "1234567",
        "dateOfBirth": "19800101",
        "additionalInformation": "19800101",
        "language": "en",
        "brummiCertificate": 1,
        "status": "accepted",
        "username": "ErMu"
        "type": "corporate"
```

```
    }
}
```

2.2 Get information about multiple drivers

URI: api/restAPIs/v1/carrier/getDrivers

2.2.1 Functional description

This method returns current available information about multiple drivers associated with the requesting company.

Data:

- *limit*: (mandatory)
  Number of drivers that should be in the response. Maximum value: 100
- *offset*: mandatory
  Defines the starting position (first position is 0). The ordering is based on the user ID with the lowest ID in position 0

2.2.2 Example data exchange

Request sent to CTC Wallet:

```
{
    "publicKey": "ctc-generated-publickey",
    "data": {
        "limit": 100,
        "offset": 0
    }
}
```

Response from CTC Wallet:

```
{
    "error_code": {
        "code": "0",
        "message": "Success"
    },
    "payload": {
        "drivers": [
            {
                "userID": 208,
                "firstName": "Erika",
                "lastName": "Musterfrau",
```

```
            "nationality": "DE",
            "primaryLanguage": null,
            "callingCode": null,
            "telephoneNumber": null,
            "dateOfBirth": "19800429",
            "additionalInformation": null,
            "language": "de",
            "brummiCertificate": 0,
            "status": "accepted",
            "username": "demo-driver@trusted-carrier.com",
            "type": "personal"
        },
        ...
        {
            <last driver>
        }
    ]
  }
}
```


2.3 Create driver

URI: api/restAPIs/v1/carrier/createDriver

2.3.1 Functional description

This method creates a new corporate driver account. The answer contains the new user ID of this driver.

For corporate driver accounts, the driver masterdata will be synchronized into the mobile app when the driver logs in into a single-user installation or into a profile of a multi-user installation.

Data:

- *username*: (mandatory)
  Must be unique on the CTC platform. Maximum of 30 characters.
- *password*: (mandatory)
  The password must have at least: 1 UPPER, 1 lower, 1 special(!@#$%^&*) and a length of at least 8 characters. The password is stored as a hashed value and is not stored in clear text in any logfiles.
- *newPasswordRequired*: (optional)
  Allowed values are “1” if driver must set a new password after the first login, or “0” if that is not required. Default is “0”.

- *language*: (optional)
  Language that will be set for this account. Value must be in ISO-639-1 format (i.e. 2-letter language code, all lower case, e.g., “de” for German) and be one of the 18 languages supported by CTC. If the field is not sent, the main company language (as defined in “Admin” module) will be used instead.
- *firstName*: (optional)
  Given name(s) of the driver. Maximum of 30 characters.
- *lastName*: (optional)
  Last name of the driver. Maximum of 30 characters.
- *nationality*: (optional)
  Nationality of the driver in ISO 3166-1 alpha-2 format (i.e. 2-letter nationality code, all upper case, e.g., “DE” for Germany)
- *primaryLanguage*: (optional)
  Primary language of the driver in ISO-639-1 format (i.e. 2-letter language code, all lower case, e.g., “de” for German)
- *dateOfBirth*: (optional)
  Birthdate of the driver (YYYYMMDD). Must be in the past.
- *callingcode*: (optional)
  Country calling code without “00” or “+” (e.g., 49 for Germany)
- *telephoneNumber*: (optional)
  Telephone number without national trunk prefix (typically a “0”), that is typically used in national calls. Only numbers are valid. Must be between 7 and 15 numbers.
- *additionalInformation*: (optional)
  Additional information about the driver. Maximum of 120 characters.
- *brummiCertificate*: (optional)
  Allowed values are “1” if you want to send a BGL-Brummi-Card certificate to the driver and “0” otherwise . This is only available for carrier companies with a confirmed membership in a BGL regional organization in the vCard service.

2.3.2 Example data exchange

Request sent to CTC Wallet:

```
{
    "publicKey": "ctc-generated-publickey",
    "data": {
        "username":"demo-driver",
        "password":"Password3#",
        "newPasswordRequired": 1,
```

```
        "firstName": "Max",
        "lastName": "Mustermann",
        "nationality": "DE",
        "primaryLanguage": "de",
        "dateOfBirth": "19801101",
        "callingCode":"49",
        "telephoneNumber": "1721234567",
        "additionalInformation": "Additional information about the driver",
        "brummiCertificate": 1
    }
}
```


Response from CTC Wallet:


```
{
    "error_code": {
        "code": "0",
        "message": "Success"
    },
    "payload": {
        "userID": 1467,
        "firstName": "Max",
        "lastName": "Mustermann",
        "nationality": "DE",
        "primaryLanguage": "de",
        "callingCode": "49",
        "telephoneNumber": "1721234567",
        "dateOfBirth": "19801101",
        "additionalInformation": "Additional information",
        "language": "de",
        "brummiCertificate": 1,
        "status": "accepted",
        "username": "demo-driver"
    }
}
```


2.4 Invite driver

URI: api/restAPIs/v1/carrier/inviteDriver

2.4.1 Functional description

This method invites a driver with a personal driver account to be associated with the company.

The workflow depends on the entered email, the role of an existing account and the current association status.

- Email is unknown: an invitation email is sent to the email address to download the mobile app CTC Wallet and create a new account. The driver can confirm the request after logging in into the mobile app. The status of the association is then “pending”.
- If the email is known on the CTC platform:
  - If the email belongs a user with a non-driver role, the API responds with an error
  - If the email belongs to a user with a personal driver account:
    - If this driver already has a confirmed association, the API responds with an a error
    - If this driver requested an association, inviteDriver will confirm the request (i.e. it will do the same as confirmDriverRequest). The status of the association is then “accepted”.
    - If this driver has no association yet, a push notification is sent to this driver. The status of the association is then “pending”.

The association request is valid for 7 days and will be automatically deleted afterwards if the driver does not react in that time period.

For personal driver accounts, the driver masterdata will not be synchronized into the mobile app.

Data:

- *email*: (mandatory)
  Email of the driver account
- *language*: (optional)
  Language that will be used in the email to invite drivers who do not have an account yet. Value must be in ISO-639-1 format (i.e. 2-letter language code, all lower case, e.g., “de” for German) and be one of the 18 languages supported by CTC. If the field is not sent, the main company language (as defined in “Admin” module) will be used instead.
- *firstName*: (optional)
  Given name(s) of the driver. Maximum of 30 characters.
- *lastName*: (optional)
  Last name of the driver. Maximum of 30 characters.
- *nationality*: (optional)
  Nationality of the driver in ISO 3166-1 alpha-2 format (i.e. 2-letter nationality code, all upper case, e.g., “DE” for Germany)

- *primaryLanguage*: (optional)
  Primary language of the driver in ISO-639-1 format (i.e. 2-letter language code, all lower case, e.g., “de” for German)
- *dateOfBirth*: (optional)
  Birthdate of the driver (YYYYMMDD). Must be in the past.
- *callingcode*: (optional)
  Country calling code without “00” or “+” (e.g., 49 for Germany)
- *telephoneNumber*: (optional)
  Telephone number without national trunk prefix (typically a “0”), that is typically used in national calls. Only numbers are valid. Must be between 7 and 15 numbers.
- *additionalInformation*: (optional)
  Additional information about the driver. Maximum of 120 characters.
- *brummiCertificate*: (optional)
  Allowed values are “1” if you want to send a BGL-Brummi-Card certificate to the driver and “0” otherwise . This is only available for carrier companies with a confirmed membership in a BGL regional organization in the vCard service.

2.4.2 Example data exchange

Request sent to CTC Wallet:

```
{
    "publicKey": "ctc-generated-publickey",
    "data": {
        "email":"demo-driver@trusted-carrier.com",
        "language": "de",
        "firstName": "Max",
        "lastName": "Mustermann",
        "nationality": "DE",
        "primaryLanguage": "de",
        "dateOfBirth": "19801101",
        "callingCode":"49",
        "telephoneNumber": "1721234567",
        "additionalInformation": "Additional information about the driver",
        "brummiCertificate": 1
    }
}
```

Response from CTC Wallet:

```
{
```

```
    "error_code": {
        "code": "0",
        "message": "Success"
    },
    "payload": {
        "userID": null,
        "firstName": "Max",
        "lastName": "Mustermann",
        "nationality": "DE",
        "primaryLanguage": "de",
        "callingCode": "49",
        "telephoneNumber": "1721234567",
        "dateOfBirth": "19801101",
        "additionalInformation": "Additional information about the driver",
        "language": "de",
        "brummiCertificate": 1,
        "status": "invited",
        "requestExpiresAt": 1702034110.216,
        "username": "demo-driver@trusted-carrier.com ",
        "type": "personal"
    }
}
```


2.5 Confirm driver request

URI: api/restAPIs/v1/carrier/confirmDriverRequest

2.5.1 Functional description

This method confirms a pending association request from users with a personal driver account. This option is available for users, if the carrier has activated it on the Wallet “Settings” page.

The status of the association changes to “accepted”.

Data:

- *userID*: (mandatory)
  User ID of the driver account. If there is no pending request from this account, the API will return an error.

2.5.2 Example data exchange

Request sent to CTC Wallet:

```
{
    "publicKey": "ctc-generated-publickey",
    "data": {
        "userID": 1128
    }
}
```


Response from CTC Wallet:


```
{
    "error_code": {
        "code": "0",
        "message": "Success"
    },
    "payload": {
        "userID": 1128,
        "firstName": "Max",
        "lastName": "Mustermann",
        "nationality": "DE",
        "primaryLanguage": "de",
        "callingCode": "49",
        "telephoneNumber": "1234567",
        "dateOfBirth": "19800101",
        "additionalInformation": "19800101",
        "language": "en",
        "brummiCertificate": 1,
        "status": "accepted",
        "username": "demo-driver@trusted-carrier.com"
        "type": "personal"
    }
}
```


2.6 Delete request

URI: api/restAPIs/v1/carrier/deleteRequest

2.6.1 Functional description

This method cancels pending association requests.

This applies to both:

- Carrier has sent an association request via email (using the frontend or using *inviteDriver*)
- Driver has sent an association request using the mobile app

Data:

- *email*: (mandatory)
  Email of the driver account. If there is no pending request, the API will return an error.

2.6.2 Example data exchange

Request sent to CTC Wallet:

```
{
    "publicKey": "ctc-generated-publickey",
    "data": {
        "email": "demo-driver@trusted-carrier.com"
    }
}
```

Response from CTC Wallet:

```
{
    "error_code": {
        "code": "0",
        "message": "Success"
    }
}
```

2.7 Remove driver

URI: api/restAPIs/v1/carrier/removeDriver

2.7.1 Functional description

This method deletes confirmed associations.

For corporate driver accounts, this deletes the association as well as the user account and all associated personal data. The user will not be able to login into the mobile app anymore.

For personal driver accounts, only the association is deleted.

Data:

- *userID*: (mandatory)
  User ID of the driver account. If there is no confirmed association, the API will return an error.

2.7.2 Example data exchange

Request sent to CTC Wallet:

```
{
    "publicKey": "ctc-generated-publickey",
    "data": {
        "userID": 1128
    }
}
```

Response from CTC Wallet:

```
{
    "error_code": {
        "code": "0",
        "message": "Success"
    }
}
```

2.8 Change password

URI: api/restAPIs/v1/carrier/changePassword

2.8.1 Functional description

This method changes the password of a corporate driver account.

For personal driver accounts, this method is not available.

Data:

- *userID*: (mandatory)
  User ID of the account
- *password*: (mandatory)
  The password must have at least: 1 UPPER, 1 lower, 1 special(!@#$%^&*) and a length of at least 8 characters. The password is stored as a hashed value and is not stored in clear text in any logfiles.
- *newPasswordRequired*: (optional)
  Allowed values are “1” if driver must set a new password after the next login, or “0” if that is not required. Default is “0”.

2.8.2 Example data exchange

Request sent to CTC Wallet:


```
{
    "publicKey": "ctc-generated-publickey",
    "data": {
        "userID": 1128,
        "password": "Password3#",
        "newPasswordRequired": 1,
    }
}
```


Response from CTC Wallet:


```
{
    "error_code": {
        "code": "0",
        "message": "Success"
    }
}
```


2.9 Remove phone number for SMS-TAN process

URI: api/restAPIs/v1/carrier/removePhoneNumber

2.9.1 Functional description

This method is to remove the phone number stored on account level of corporate driver accounts. The phone number is used for receiving SMS-TANs which are needed to login into a profile of a Multi-user installation of the mobile app.

Users with personal driver accounts can reset the phone number using their email address.

Data:

- *userID*: (mandatory)
  User ID of the account

2.9.2 Example data exchange

Request sent to CTC Wallet:


```
{
    "publicKey": "ctc-generated-publickey",
    "data": {
```

```
        "userID": 1128
    }
}
```


Response from CTC Wallet:


```
{
    "error_code": {
        "code": "0",
        "message": "Success"
    }
}
```


2.10 Update driver information

URI: api/restAPIs/v1/carrier/updateDriver

2.10.1 Functional description

This method is used to update information about a driver with a pending or confirmed association.

If a driver requested an association, calling this method confirms the driver's request (i.e. it does the same as *confirmDriverRequest*)

Data:

- *userID*: (mandatory)
  User ID of the account
- *username*: (optional)
  User name can be changed (only for corporate driver accounts). Maximum of 30 characters.
- All other fields: see *createDriver* / *inviteDriver* methods.

If a field is not sent, the existing information is preserved.

To delete the existing information of a field, an empty string ("") must be sent. Fields that cannot be deleted: userID, username, language

2.10.2 Example data exchange

Request sent to CTC Wallet:


```
{
    "publicKey": "ctc-generated-publickey",
    "data": {
        "userID":1128,
        "username":"demo-driver2",
        "additionalInformation": "",
        "brummiCertificate": 1
    }
}
```


Response from CTC Wallet:


```
{
    "error_code": {
        "code": "0",
        "message": "Success"
    },
    "payload": {
        "userID": 1128,
        "language": "de",
        "firstName": "Max",
        "lastName": "Mustermann",
        "nationality": "DE",
        "primaryLanguage": "de",
        "callingCode": "49",
        "telephoneNumber": "1721234567",
        "dateOfBirth": "19801101",
        "additionalInformation": null,
        "brummiCertificate": 1,
        "status": "accepted",
        "username": "demo-driver2"
    }
}
```


3 Methods to manage driver authentication in Multi-User installations

Access to the methods in this chapter will only be granted to carriers who are able to document a secure way of identifying & authenticating the driver.

Wording clarification inside this chaper:

- “Login” / “Logout” is used for processes involving the driver entering their profile on a CTC Wallet installation running in Multi User mode
- “MFA-Login” / “MFA-Logout” is used for processes involving the driver authenticating in a 3rd party device/app outside the CTC ecosystem using a personal authentication method.

Specific error codes

- {"code":"5011","message":"TC API usage for 2-factor-authentication is not configured or has yet to be approved"}
  Carrier did not configure to use TC API, or the use was not approved by CTC
- {"code":"5012","message":"The internal driver ID is unknown"}
  Internal driver ID is not associated with any driver
- {"code":"5013","message":"The deviceID is unknown"}
  Device ID is not associated with any MU installation
- {"code":"5014","message":"Invalid timestamp"}
  Timestamp of the login event must be in the past, but not older than 24 hours
- {"code":"5015","message":"Driver is not logged on the specified device"}
  deviceID does not match driverID in database of mfaLogin events

3.1 MFA Login event

URI: api/restAPIs/v1/carrier/mfaLogin

3.1.1 Functional description

The call for the MFA-Login is triggered by the driver confirming their identity using a personal authentication method. This may include:

- An action that uses a personalized token (e.g., slot-in event of driver card into the digital tachograph)
- Using a password known only to the driver and logging in into a 3rd party device/app

The call is not intended to affect the login status of a driver on a CTC Wallet installation running in Multi User mode. The call will therefore not automatically login the driver from an MU device. The MFA-login information is stored as a tuple (internal driver ID, deviceID) in the CTC backend. When a driver logs into their driver profile, the CTC backend will be queried for the tuple.

There cannot be more than 1 tuple in the CTC backend that has the same internal driver ID. There cannot be more than 1 tuple in the CTC backend that has the same device ID.

Data:

- *internalDriverID*: (mandatory)
  Internal Driver ID must be associated with an existing driver association
- *deviceID*: (mandatory)
  Device ID must be a known MU installation device name
- *timestamp*: (mandatory)
  Timestamp in unixtime. Must be in the past, but not older than 24 hours

3.1.2 Example data exchange

Request sent to CTC Wallet:

```
{
    "publicKey": "ctc-generated-publickey",
    "data": {
        "internalDriverID": "A00123",
        "deviceID": "TRUCK-D12",
        "timestamp": 1714387156
    }
}
```

Response from CTC Wallet:

```
{
    "error_code": {
        "code": "0",
        "message": "Success"
    }
}
```

3.2 MFA Logout event

URI: api/restAPIs/v1/carrier/mfaLogout

3.2.1 Functional description

The call for the MFA-Logout can be triggered by a session-ending event in a 3rd party device/app. This may include:

- An action that uses a personalized token (e.g., slot-out event of driver card)

- Logging out of a 3rd party device/app (manual or time-based)

Assuming a matching Multi User mode deviceID, the call will also automatically log out the driver on this CTC Wallet installation, otherwise an error will be returned.

MFA-logout also removes the internal driver ID from an existing tuple in the CTC backend. However, the driver will remain able to login into the driver profile on this device, as long as they are the last logged in driver.

The field “mfaLoginTimestamp” is used to identify the correct MFA-login event. The API call will return an error if the field “mfaLoginTimestamp” doesn’t match the latest MFA-login timestamp and the user will not be logged out. This is done to prevent accidental logout events.

Data:

- *internalDriverID*: (mandatory)
  Internal Driver ID must be associated with an existing driver association
- *deviceID*: (mandatory)
  Device ID must be a known MU installation device name
- *timestamp*: (mandatory)
  Timestamp in unixtime. Must be in the past, but not older than 24 hours

3.2.2 Example data exchange

Request sent to CTC Wallet:

```
{
    "publicKey": "ctc-generated-publickey",
    "data": {
        "internalDriverID": "A00123",
        "deviceID": "TRUCK-D12",
        "mfaLoginTimestamp": 1714387156
    }
}
```

Response from CTC Wallet:

```
{
    "error_code": {
        "code": "0",
        "message": "Success"
    }
```

}

4 Notifications to external systems

Notifications are only sent if an endpoint and a key are entered in the “Admin” module.

Changes of the status of an association are only possible for associations with drivers with a “Personal” account.

4.1 Driver accepts association request by the carrier

4.1.1 Functional description

When the carrier sends an association request to the driver (via frontend or API), the driver receives an email (if they do not have an account yet) or a push notification into their mobile app (if they already have an account).

Here, the driver **accepts** the association request, thus setting the association status to “accepted”.

Note: The same notification is sent if a driver logs in into a Multi-user installation without having a prior association with the carrier.

4.1.2 Example data exchange

Notification sent by CTC Wallet:


```
{
    "publicKey": "ERP-publickey",
    "type": "wallet:driver",
    "payload": {
        "userID": 1128,
        "firstName": "Max",
        "lastName": "Mustermann",
        "nationality": "DE",
        "primaryLanguage": "de",
        "callingCode": "49",
        "telephoneNumber": "1234567",
        "dateOfBirth": "19800101",
        "additionalInformation": "19800101",
        "language": "en",
        "brummiCertificate": 1,
        "status": "accepted",
        "username": "demo-driver@trusted-carrier.com"
```

```
        "type": "personal"
    }
}
```

Expected response:

```
{
    "error_code": {
        "code": "0",
        "message": "Success"
    }
}
```

4.2 Driver rejects association request by the carrier

4.2.1 Functional description

When the carrier sends an association request to the driver (via frontend or API), the driver receives an email (if they do not have an account yet) or a push notification into their mobile app (if they already have an account).

Here, the driver **rejects** the association request, thus setting the association status to “removed” and deleting the association with all its associated data. The driver will not appear in future calls of *getDriver* / *getDrivers*.

4.2.2 Example data exchange

Notification sent by CTC Wallet:

```
{
    "publicKey": "ERP-publickey",
    "type": "wallet:driver",
    "payload": {
        "userID": 1128,
        "firstName": null,
        "lastName": null,
        "nationality": null,
        "primaryLanguage": null,
        "callingCode": null,
        "telephoneNumber": null",
        "dateOfBirth": null,
        "additionalInformation": null,
        "language": null,
        "brummiCertificate": 0,
```

```
        "status": "removed",
        "username": "demo-driver@trusted-carrier.com"
        "type": "personal"
    }
}
```


Expected response:


```
{
    "error_code": {
        "code": "0",
        "message": "Success"
    }
}
```


4.3 Driver removes an existing association

4.3.1 Functional description

When there is a confirmed association between a carrier and a driver, both have the option to delete the association.

When the driver deletes the association, the association status is set to “removed” and all its associated data is deleted. The driver will not appear in future calls of *getDriver* / *getDrivers*.

Note: The same notification is sent if a driver account is deleted, which then also deletes the association.

4.3.2 Example data exchange

Notification sent by CTC Wallet:


```
{
    "publicKey": "ERP-publickey",
    "type": "wallet:driver",
    "payload": {
        "userID": 1128,
        "firstName": null,
        "lastName": null,
        "nationality": null,
        "primaryLanguage": null,
        "callingCode": null,
        "telephoneNumber": null",
        "dateOfBirth": null,
        "additionalInformation": null,
        "language": null,
```

```
        "brummiCertificate": 0,
        "status": "removed",
        "username": "demo-driver@trusted-carrier.com"
        "type": "personal"
    }
}
```


Expected response:


```
{
    "error_code": {
        "code": "0",
        "message": "Success"
    }
}
```


4.4 Driver requests an association

4.4.1 Functional description

Drivers can request an association with a carrier, if the carrier has chosen to allow this on the “Settings” page in the “Wallet” module.

If there already is a pending association by the carrier for the same driver, the association request is automatically confirmed. I.e. the same notification as in chapter 5.1. is sent.

If there is no pending association, an association request with status “driverRequested” is created and a notification is sent. The association request is valid for 7 days and will be automatically deleted afterwards if the carrier does not react in that time period.

4.4.2 Example data exchange

Notification sent by CTC Wallet:


```
{
    "publicKey": "ERP-publickey",
    "type": "wallet:driver",
    "payload": {
        "userID": 1128,
        "primaryLanguage": null,
        "callingCode": null,
        "telephoneNumber": null,
        "dateOfBirth": null,
        "status": "driverRequested",
        "language": "en",
        "brummiCertificate": 0,
```

```
        "expiresAt": 1700054744.949,
        "username": "demo-driver@trusted-carrier.com",
        "type": "personal"
    }
}
```


Expected response:


```
{
    "error_code": {
        "code": "0",
        "message": "Success"
    }
}
```


4.5 Driver deletes certificate issued by the carrier

4.5.1 Functional description

When a driver deletes a certificate issued by the carrier (e.g., “BGL-Brummi-Card” certificate that is available for BGL-certified carriers), a notification is sent with the updated information.

4.5.2 Example data exchange

Notification sent by CTC Wallet:


```
{
    "publicKey": "ERP-publickey",
    "type": "wallet:driver",
    "payload": {
        "userID": 1128,
        "firstName": "Max",
        "lastName": "Mustermann",
        "nationality": "DE",
        "primaryLanguage": "de",
        "callingCode": "49",
        "telephoneNumber": "1234567",
        "dateOfBirth": "19800101",
        "additionalInformation": "19800101",
        "language": "en",
        "brummiCertificate": 0,
        "status": "accepted",
        "username": "demo-driver@trusted-carrier.com"
```

```
        "type": "personal"
    }
}
```

Expected response:

```
{
    "error_code": {
        "code": "0",
        "message": "Success"
    }
}
```