Cardnumber validation

GET /patron/cardnumber/{cardno}/library/{libraryid}

This endpoint only validates the subscription of a given patron library card. For security reasons this endpoint does not return a patronSystemId.

Cardnumber lookup

GET /library/{libraryid}/patron?offset=0&limit=10&search={cardno}

To search for a patron by cardnumber this endpoint will return a patronSystemId. To use this endpoint the API-user needs to have permissions for patron administration. The endpoint returns a patronSystemId which can be used in other requests to get patron information.

Patron information

GET /patron/{patronid}/library/{libraryid}/patroninformation

This endpoint returns the basic patron information such as name, address and subscription information. The patron must be logged in to get this information.

External customers (CRM)

Wise can act as database for customers from external applications. For example a ticketing application can register customer information in Wise. The customer is stored with an role “XTRN” in Wise. The application can retrieve and update the information at a later time. When the customer also has a library subscription, then the existing patron will be used and the XTRN role will be added to his/her record.

Storing external customers

Before storing a customer, the application must check whether the customer already exists.

POST /patron/external/library/{libraryid}
Subscribing patrons

From Wise version 6.6.5 onwards the subscription of patrons via an endpoint is supported.

The following endpoints are available to complete the subscription process:

- /subscription/{subscriptionType}/instance/{instanceId}/field
- /library/{libraryId}/subscription?branchId=branchId&patronSystemId=patronSystemId &dateOfBirth=yyyy-mm-dd&gender=gender&zipCode=zipcode
- /patron/authentication/library/{instanceId}
- /patron/authentication/
- /library/{libraryId}/patron
- /library/{libraryId}/patron/{patronSystemId}
- /payment/{paymentId}/status/{paymentStatus}

The process needs to follow the following steps:

1. Get the possible subscriptions for the new patron. The offered subscriptions can be age-dependent and the fees can differ from region to region.
2. If the patron is already known, see if a login with the patron’s credentials leads to a success or failure. The outcome determines whether the subscription needs to be done with a POST or a PUT with a given patron system id.
3. Depending on option VWIZBET the subscription is activated with a status ‘invoiced’ or added to the shopping basket that needs to be paid prior to activating the new subscription. If the subscription product is in the shopping basket the basket can be paid via the /payment-endpoint.

Get required fields

This is an optional endpoint that can be called to get information about which fields are mandatory in the subscription process. This can be used to inform the new patron about the fields that needs to be provided before the subscription can be completed.

Type: GET

Request: subscription/R/instance/{libraryId}/field

Parameters

- LibraryId: this is the id (instantie_id) from the library the patron wants to (re)subscribe to

Response:

- Id: id that identifies the field, like ‘PML’ stands for ‘Email address’
- Type: C(haracter), N(umeric), D(ate)
- Label: field name how it can be presented for the patron
- Required: Boolean, is this field required to provide
• Desired: Boolean, not required but nice to store
• Visible: Boolean, should this field be presented to the patron
• Description: not applicable, always empty
• Format: how should the patron enter the data
• Group: an indicator that can group fields together
• GroupLabel: a description of the group. Can be used to bind fields in a group together on the screen
• Order: in what order should fields be presented on the screen.

Sample:

[
{
'id': 'PML', 'validationErrors': [], 'type': 'C', 'label': 'Emailaddress',
'required': True, 'desired': False, 'visible': True, 'description': '', 'format':
'email', 'group': 'A', 'groupLabel': 'APP-FORMS__LABEL-AUTHENTICATION', 'order':
'12-1'},
{
'id': 'PGN', 'validationErrors': [], 'type': 'C', 'label': 'Username', 'required':
True, 'desired': False, 'visible': True, 'description': '', 'format': '', 'group':
'A', 'groupLabel': 'APP-FORMS__LABEL-AUTHENTICATION', 'order': '13-1'},
...
]

At least the following fields are checked by the subscription endpoint and need to be provided:

• STREET
• CITY
• LAST_NAME
• DATE_OF_BIRTH
• HOUSE_NUMBER
• HOME_BRANCH
• USERNAME
• PASSWORD
• ZIPCODE
• GENDER
• EMAIL_ADDRESS

Get possible subscriptions

To get the possible subscriptions for the new or re-subscribing patron, Wise offers the subscription endpoint. The endpoint responds only with the subscription which are available via web.
Type: GET

Request:
/library/{libraryId}/subscription?branchId=branchId&patronSystemId=patronSystemId&dateOfBirth=yyyy-mm-dd&gender=gender&zipCode=zipcode

Parameters:

- **LibraryId**: this is the id (instantie_id) from the library the patron wants to (re)subscribe to.
- **BranchId**: this is the actual branch of the library the patron wants to (re)subscribe to. Because every branch can have its own fees for subscriptions, this branch-id is important to provide.
- **Date of birth**: In Wise, subscriptions can be configured for specific age-groups. Only the subscriptions applicable for the (new) patron are offered.
- **Gender**: FEMALE or MALE. Only natural patrons can subscribe via the rest-endpoint. Organizations cannot subscribe.
- **Zipcode**: within the branchId some zipcodes can offer subscriptions for a reduced or raised price.

Response:

```json
{
  "subscriptionId": 0,
  "subscriptionDescription": "string",
  "cost": "string",
  "renewalType": "string",
  "subscriptionCost": 0,
  "subscriptionType": "string",
  "subscriptionLength": 0,
  "internetFee": 0,
  "additionalInformation": "string",
  "requiresCAPTCHA": true
}
```

- **SubscriptionId**: the id of the subscription that should be used when the subscription endpoint is called.
- **SubscriptionDescription**: a general description like ‘Youth 10-15’ or ‘Standard 18-99’ etc.
- **Cost**: the cost of the subscription in text, like “€ 21.00 /12 months”
- **renewalType**: not applicable for rest-API subscription
- **SubscriptionCost**: the cost of the subscription in cents, like “2100”
• SubscriptionType: more information about the subscription-group, like “18-99”
• SubscriptionLength: number of months for this subscription (12 = a year subscription)
• InternetFee: minimal amount to be paid when subscribing via the internet
• AdditionalInformation: can contain more information about the subscription.
• RequiresCAPTCHA: if a captcha is required this can be set to 'true' in the klanttype_velden-table.

Sample:

```json
[{"subscriptionId":45,"subscriptionDescription":"Standaardsubscription","cost":"€ 43.50 /12 mnd","renewalType":"0","subscriptionCost":4350,"subscriptionType":"18-64","subscriptionLength":12,"internetFee":0,"additionalInformation":"Default subscription. incl. Ebooks","requiresCAPTCHA":false},
{"subscriptionId":200,"subscriptionDescription":"Digital Standard (Ebooks)","cost":"€ 42.00 /12 mnd","renewalType":"0","subscriptionCost":4200,"subscriptionType":"18-99","subscriptionLength":12,"internetFee":0,"additionalInformation":"Digital Standard (Ebooks)","requiresCAPTCHA":false}]
```

Authenticate patron

There are two endpoints that can be used to authenticate a patron:

- `/patron/authentication/library/{instanceId}`
- `/patron/authentication/`

The first endpoint will check whether the patron has an active subscription in the given library-id.

In Flanders a patron can have more than one active subscription in multiple libraries. This is not the case in other Wise installations. Flanders will make use of the first mentioned endpoint.

Type: POST

Request: `patron/authentication/library/{libraryId}`

Data:

"username": username,
"password": password

Username can be the patron’s username or a barcode

The same call can be done without a specific library:

Request: `patron/authentication/`

Response:
• LibraryId: the id of the library where the patron is logged in
• PatronSystemId: a unique id for this patron. This patronSystemId is needed to resubscribe the patron
• MembershipexpiryDate: the date the current active subscription ends. YYYY-MM-DD
• Token: this token is needed to resubscribe the patron and should be used in the header of the resubscribe endpoint
• Username: contains the username or barcode

Sample:

{"id":null,"validationErrors":null,"libraryId":"I018","patronSystemId":"1328ebb152edc752328b79d37b553bea","membershipExpiryDate":"2022-05-31","token":null,"username":null}

Subscribe patron

There are two endpoints for subscribing a patron. The first one is used for new patrons and the second one for subscribing patrons in a new library or re-subscribing patrons in libraries where they previously had a membership.

The outcome of the login endpoint determines which one should be used.

If the ‘authenticate’ patron does not return a patronSystemId then the first one (/library/{libraryId}/patron) should be used, otherwise the second one should be used (/library/{libraryId}/patron/{patronSystemId}).

Re-subscription of patrons can only be done if the patron has no active subscription in the given library or when the subscription is ending at the end of the current month.

Method 1: new patrons

Type: POST

Request: /library/{libraryId}/patron

Parameters

• LibraryId: this is the id (instantie_id) from the library the patron wants to subscribe

Data:

{
  "initials": "",
  "firstName": "",
  "lastName": "",
  "dateOfBirth": "",
  "dateOfDeath": ""
}
{
  "addresses": [
    {
      "street": "",
      "houseNumber": "",
      "addition": "",
      "zipCode": "",
      "city": "",
      "region": "",
      "country": "",
      "type": "H",
      "typeLabel": "",
      "method": ""
    }
  ],
  "phoneNumber": "",
  "emailAddress": "",
  "homeBranch": "",
  "identificationNumber": {
    "identificationType": "EID",
    "identificationNumber": ""
  },
  "preferredLanguage": "",
  "patronSystemId": ""
  "password": "",
  "username": ""
  "gender": ""
"smsNumber": "",
"futureSubscriptionId": "",
"acceptedTermsAndAgreement": "true"
}

The bold lines are mandatory and if not supplied will lead to an error.

Field "homeBranch" is the branch in which the patron wants a new subscription. If this branch is not one of the libraries from the library-id in the parameter a 500-error will be thrown.

Response:

Most of the response is self-explanatory:

- PatronId: contains the new generated actor_id.
- PatronSystemId: contains the new generated unique id for this patron.
- SubscriptionStatus: the current subscription status for this new patron.
- PaymentId: a unique identifier that can be used to pay the subscription via Ideal/Rest if the status is not 'INVOICE'.

Sample:

```json
{"id": ","validationErrors": [],"patronId": 1516143,"patronSystemId": "9d6d1e8ae0a47b43988971ffa4834126","initials": "T.","middleName": "","customerTag": "","dateOfBirth": "1980-01-01","showLoanHistory": "B"}
```

Method 2: re-subscribe patrons

Type: PUT

Request: /library/{libraryId}/patron/{patronSystemId}

Parameters

- LibraryId: this is the id (instantie_id) from the library the patron wants to subscribe
- PatronSystemId: this is the id that was obtained by the 'authorization' request (patron/authentication/library/{libraryId})

Data:

```json
{}
```

"initials": "",
"firstName": "",
"lastName": "",
"dateOfBirth": "",
"addresses": [
{
"street": "",
"houseNumber": "",
"addition": "",
"zipCode": "",
"city": "",
"region": "",
"country": "",
"type": "H",
"typeLabel": "",
"method": ""
}
],
"phoneNumber": "",
"emailAddress": "",
"homeBranch": "",
"identificationNumber": {
"identificationType": "EID",
"identificationNumber": ""
},
"preferredLanguage": "",
"patronSystemId": ""
"password": "",
"username": ""
"gender": "",
"smsNumber": "",
"subscriptionId": "",
"acceptedTermsAndAgreement": "true"
}

The bold lines are mandatory and will lead to an error if not supplied. Please note that now patronSystemId is also mandatory.

One of the requirements to subscribe a patron is that in the subscriptions-table in Wise for every library a definition for customer-type ‘N’ (anonymous) is available. Without this definition a subscription cannot be completed. See also the regular Wise manuals for more information about subscriptions.

**Pay subscription fees**

If the option VWIZBET is not set in Wise a subscription will not be activated and will remain in the shopping basket until the payment is completed. The payment-id from the subscription endpoint should be used to complete the subscription.

Type: POST

Request: /payment/{paymentId}/status/{paymentStatus}

Parameters:

- PaymentId: the id that came in the response from the subscription endpoint
- PaymentStatus: SUCCESS. For other statuses, see the swagger documentation ([Swagger UI (hostedwise.nl)]).

Response:

A default message code 200 is given if the payment is handled successfully.

Before a client can call this endpoint, the client needs to authorize itself with a special account (username and password) provided by the library organization. It is required that the ip-number of the client is added to the grantlist.

The token in the response is needed to complete the payment and should be used in the header of the payment endpoint.

**Retrieve blocks for a specific patron**

With this endpoint the current blocks for a patron can be retrieved.
**Type: GET**

**Request:**

`/patronmanagement/patron/{patronSystemId}/library/{libraryId}/property/blocking`

**Parameters**

- **PatronSystemId:** this is the id that was obtained by the 'authorization' request (patron/authentication/library/{libraryId})
- **LibraryId:** this is the id (instantie_id) from the library where the block is active.

**Response:**

```json
{
  "id": "",
  "validationErrors": [],
  "addressBlock": false,
  "generalBlock": false,
  "onlineBlock": false,
  "customerAtTerminalBlock": false,
  "webPortalSelfServiceBlock": false,
  "collectionAgencyBlock": false,
  "patronCardBlock": false,
  "blockMessage": "",
  "blockDetails": ""
}
```

**Set or remove one or more blocks for a patron**

With this endpoint it is made possible to set or remove a few blocks on a specific patron.

**Type: PUT**

**Request:**

`/patronmanagement/patron/{patronSystemId}/library/{libraryId}/property/blocking`

- **PatronSystemId:** this is the id that was obtained by the 'authorization' request (patron/authentication/library/{libraryId})
- **LibraryId:** this is the id (instantie_id) from the library where the block should be activated or deactivated
Set the required type of block on ‘true’ to activate the block. Set it to ‘false’ to deactivate it.

In the response the current situation is returned.

Response:

```json
{
    "id": "",
    "validationErrors": [],
    "addressBlock": false,
    "generalBlock": false,
    "onlineBlock": false,
    "customerAtTerminalBlock": false,
    "webPortalSelfServiceBlock": false,
    "collectionAgencyBlock": false,
    "patronCardBlock": false,
    "blockMessage": "",
    "blockDetails": ""
}
```
Before this call can be done the client needs to be logged in. The client needs at least ‘read’ permissions on the ‘actoren’, on library-level.