SSE/14789/IFS/0002: CMS Integration Services

The Integration Services interface for Cicero LMS is designed to be used by the end-user interfaces of a library, for users to be able to check up on loans, bills and more. A prerequisite for using this interface is that the client application must be user-faced, and have an interactive flow with the user that only fetches information from the interface when it is needed.

This interface must not be used for other purposes than what is described here. Examples of detrimental usage:

- Fetching large amounts of information, e.g. to harvest information from patrons or libraries
- Fetching patron information when the patron is not using the system
- Fetching library information when it is not used in conjunction with the user using the system
- Other activities which do not adhere to the described purpose

Detrimental usage of the CMS interface can lead to the client application getting its access to the interface closed down.

It is generally allowed to have a local cache with information. Updating the local cache must however be part of the normal user scenarios of the system - information harvesting, which for a shorter or a longer period of time makes many requests to the CMS interface, will not be tolerated.

Some operations are suited to having information cached across patrons in the application. This information is limited to:

- Configurations
- LoanerGroups
- Operations for fetching the placements of the library (Branches, Departments, Locations and Sublocations)

Catalog

Show/Hide | List Operations | Expand Operations | Raw

GET /external/{agencyid}/catalog/availability/v3 — Get availability of bibliographical records.

Implementation Notes

Returns an array of availability for each bibliographical record.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
recordid	**list of record ids**	query	array[string]
exclude	Identifies the branchIds which are excluded from the result	query	array[string]

Response Class

Model | Model Schema

AvailabilityV3 {
  recordId (string): The FAUST number of the Bibliographic record,
  reservations (integer): Total number of current active reservations of the Bibliographic record,
  reservable (boolean): True if materials can be reserved,
  available (boolean): True if materials is available on-shelf at some placement, false if all materials are lent out
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/catalog/bookingInformation/{recordid}/v3

Retrieves all booking information for each branch of an agency for bookings that ends after the current date.

Implementation Notes

Returns an array of BookingBranchInfo which contains:

- the branch ID
- gross number of available materials for that branch
- array of BookingInfo for the given bibliographic record, containing the booking Period and the number of preferred materials

This is to highlight when materials are available for a new potential booking.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
recordid	**identifies the bibliographical record, i.e. the FAUST number**	path	string

Response Class

Model | Model Schema

BookingBranchInfoV3 {
branchId (string): The branch ID,
grossNumberAvailable (integer): The gross number of available materials,
bookingInfo (array[BookingInfo]): Details about requested booking information
}
BookingInfo {
preferredMaterials (integer): The preferred number of materials,
period (Period): The booking period information containing the start and the end date
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request com.dantek.dl.rest.RestException	
401	client unauthorized	
404	patron not found	

GET /external/{agencyid}/catalog/holdings/v3

Get placement holdings for bibliographical records.

Implementation Notes

This method has been deprecated. Consider using /external/{agencyid}/catalog/holdings/v5 instead Returns an array of holdings for each bibliographical record. The holdings lists the materials on each placement, and whether they are available on-shelf or lent out.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
recordid	**Identifies the bibliographical records - The FAUST number.**	query	array[string]
exclude	Identifies the branchIds which are excluded from the result	query	array[string]

Response Class

Model | Model Schema

HoldingsForBibliographicalRecordV3 {

recordId (string): Identifies the bibliographical record for the available materials, The FAUST number,
reservations (integer): Total number of current active reservations for the bibliographical record,
reservable (boolean): True if there is any reservable materials,
holdings (array[HoldingsV3]): An array of holdings for the materials matching the bibliographical record, as distributed across branches, departments and locations

```
}
HoldingsV3 {
   materials (array[MaterialV3]): Materials that belongs to this placement,
   location (AgencyLocation, optional): Placement location,
   sublocation (AgencySublocation, optional): Placement sublocation,
   department (AgencyDepartment, optional): Placement department,
   branch (AgencyBranch): Placement branch
}
MaterialV3 {
   itemNumber (string): Identifies the material,
   materialGroup (MaterialGroup): Name of the material group that the material belongs to,
   periodical (Periodical, optional): Present if material is a periodical,
   available (boolean): True if material is available on-shelf, false if lent out
}
MaterialGroup {
   name (string): Name of the material group,
   description (string, optional): Description of the material group
}
Periodical {
   volume (string, optional),
   volumeYear (string, optional),
   displayText (string): A representation of the periodica volume information that is suitable for display,
   volumeNumber (string, optional)
}
AgencyLocation {
   locationId (string): Location identifier,
   title (string): Name of the location
}
AgencySublocation {
   title (string): Name of the sub-location,
   sublocationId (string): Sub-location identifier
}
AgencyDepartment {
   departmentId (string): Department identifier,
   title (string): Name of the department
```

}

AgencyBranch {
  branchId (string): ISIL of branch (e.g. DK-761501),
  title (string): Name of branch
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/catalog/holdings/v4 Get placement holdings for bibliographical records.

Implementation Notes

Returns an array of holdings for each bibliographical record. The holdings lists the materials on each placement, and whether they are available on-shelf or lent out.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
recordid	**Identifies the bibliographical records - The FAUST number.**	query	array[string]
exclude	Identifies the branchIds which are excluded from the result	query	array[string]

Response Class

Model | Model Schema

HoldingsForBibliographicalRecordV4 {
  recordId (string): Identifies the bibliographical record for the available materials, The FAUST number,

reservations (integer): Total number of current active reservations for the bibliographical record,
reservable (boolean): True if there is any reservable materials,
holdings (array[HoldingsV4]): An array of holdings for the materials matching the bibliographical record, as distributed across branches, departments and locations
}
HoldingsV4 {
materials (array[MaterialV3]): Materials that belongs to this placement,
location (AgencyLocation, *optional*): Placement location,
sublocation (AgencySublocation, *optional*): Placement sublocation,
section (AgencySection, *optional*): Placement section,
department (AgencyDepartment, *optional*): Placement department,
branch (AgencyBranch): Placement branch
}
MaterialV3 {
itemNumber (string): Identifies the material,
materialGroup (MaterialGroup): Name of the material group that the material belongs to,
periodical (Periodical, *optional*): Present if material is a periodical,
available (boolean): True if material is available on-shelf, false if lent out
}
MaterialGroup {
name (string): Name of the material group,
description (string, *optional*): Description of the material group
}
Periodical {
volume (string, *optional*),
volumeYear (string, *optional*),
displayText (string): A representation of the periodica volume information that is suitable for display,
volumeNumber (string, *optional*)
}
AgencyLocation {
locationId (string): Location identifier,
title (string): Name of the location
}
AgencySublocation {
title (string): Name of the sub-location,

sublocationId (string): Sub-location identifier
}
AgencySection {
sectionId (string): Section identifier,
title (string): Name of the section
}
AgencyDepartment {
departmentId (string): Department identifier,
title (string): Name of the department
}
AgencyBranch {
branchId (string): ISIL of branch (e.g. DK-761501),
title (string): Name of branch
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/catalog/holdings/v5

Get placement holdings for bibliographical records.

Implementation Notes

Returns an array of holdings for each bibliographical record. The holdings lists the materials on each placement, and whether they are available on-shelf or lent out. If the materials are lent out, a return date will also be available.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string

Parameter	Description	Parameter Type	Data Type
recordid	**Identifies the bibliographical records - The FAUST number.**	query	array[string]
exclude	Identifies the branchIds which are excluded from the result	query	array[string]

Response Class

Model | Model Schema

HoldingsForBibliographicalRecordV5 {
  recordId (string): Identifies the bibliographical record for the available materials, The FAUST number,
  reservations (integer): Total number of current active reservations for the bibliographical record,
  reservable (boolean): True if there is any reservable materials,
  holdings (array[HoldingsV5]): An array of holdings for the materials matching the bibliographical record, as distributed across placements
}
HoldingsV5 {
  materials (array[MaterialV4]): Materials that belongs to this placement,
  location (AgencyLocation, *optional*): Placement location,
  sublocation (AgencySublocation, *optional*): Placement sublocation,
  section (AgencySection, *optional*): Placement section,
  department (AgencyDepartment, *optional*): Placement department,
  branch (AgencyBranch): Placement branch
}
MaterialV4 {
  itemNumber (string): Identifies the material,
  returnDate (string): Return date of the material if it is lendout. Format: yyyy-mm-dd,
  materialGroup (MaterialGroup): Name of the material group that the material belongs to,
  periodical (Periodical, *optional*): Present if material is a periodical,
  available (boolean): True if material is available on-shelf, false if lent out
}
MaterialGroup {
  name (string): Name of the material group,
  description (string, *optional*): Description of the material group
}
Periodical {

```
  volume (string, optional),
  volumeYear (string, optional),
  displayText (string): A representation of the periodica volume information that is suitable for display,
  volumeNumber (string, optional)
}
AgencyLocation {
  locationId (string): Location identifier,
  title (string): Name of the location
}
AgencySublocation {
  title (string): Name of the sub-location,
  sublocationId (string): Sub-location identifier
}
AgencySection {
  sectionId (string): Section identifier,
  title (string): Name of the section
}
AgencyDepartment {
  departmentId (string): Department identifier,
  title (string): Name of the department
}
AgencyBranch {
  branchId (string): ISIL of branch (e.g. DK-761501),
  title (string): Name of branch
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
`400`	bad request	
`401`	client unauthorized	

GET /external/{agencyid}/catalog/holdingsLogistics/v1 Get placement holdings for bibliographical records.

Implementation Notes

Returns an array of holdings for each bibliographical record. The holdings lists the materials on each placement, and whether they are available on-shelf or lent out.

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**ISIL of the agency (e.g. DK-761500)**	path	string
`recordid`	**Identifies the bibliographical records - The FAUST number.**	query	array[string]
`exclude`	Identifies the branchIds which are excluded from the result	query	array[string]

Response Class

Model | Model Schema

HoldingsForBibliographicalRecordLogisticsV1 {
recordId (string): Identifies the bibliographical record for the available materials, The FAUST number,
reservations (integer): Total number of current active reservations for the bibliographical record,
reservable (boolean): True if there is any reservable materials,
holdings (array[HoldingsLogisticsV1]): An array of holdings for the materials matching the bibliographical record, as distributed across branches, departments and locations
}
HoldingsLogisticsV1 {
materials (array[MaterialV3]): Materials that belongs to this placement,
lmsPlacement (PlacementV1, *optional*): LMS placement,
branch (AgencyBranch): Placement branch,
logisticsPlacement (array[string], *optional*): Logistics placement
}
MaterialV3 {
itemNumber (string): Identifies the material,
materialGroup (MaterialGroup): Name of the material group that the material belongs to,
periodical (Periodical, *optional*): Present if material is a periodical,

available (boolean): True if material is available on-shelf, false if lent out
}
MaterialGroup {
name (string): Name of the material group,
description (string, *optional*): Description of the material group
}
Periodical {
volume (string, *optional*),
volumeYear (string, *optional*),
displayText (string): A representation of the periodica volume information that is suitable for display,
volumeNumber (string, *optional*)
}
PlacementV1 {
section (AgencySection, *optional*): Placement section,
location (AgencyLocation, *optional*): Placement location,
sublocation (AgencySublocation, *optional*): Placement sublocation,
department (AgencyDepartment, *optional*): Placement department
}
AgencySection {
sectionId (string): Section identifier,
title (string): Name of the section
}
AgencyLocation {
locationId (string): Location identifier,
title (string): Name of the location
}
AgencySublocation {
title (string): Name of the sub-location,
sublocationId (string): Sub-location identifier
}
AgencyDepartment {
departmentId (string): Department identifier,
title (string): Name of the department
}
AgencyBranch {

branchId (string): ISIL of branch (e.g. DK-761501),
title (string): Name of branch
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/catalog/holdingsLogistics/v2

Get placement holdings for bibliographical records.

Implementation Notes

Returns an array of holdings for each bibliographical record. The holdings lists the materials on each placement, and whether they are available on-shelf or lent out. If the materials are lent out, a return date will also be available.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
recordid	**Identifies the bibliographical records - The FAUST number.**	query	array[string]
exclude	Identifies the branchIds which are excluded from the result	query	array[string]

Response Class

Model Model Schema

HoldingsForBibliographicalRecordLogisticsV2 {
recordId (string): Identifies the bibliographical record for the available materials, The FAUST number,
reservations (integer): Total number of current active reservations for the bibliographical record,
reservable (boolean): True if there is any reservable materials,

```
    holdings (array[HoldingsLogisticsV2]): An array of holdings for the materials matching the bibliographical record, as distributed across placements
}
HoldingsLogisticsV2 {
    materials (array[MaterialV4]): Materials that belongs to this placement,
    lmsPlacement (PlacementV1, optional): LMS placement,
    branch (AgencyBranch): Placement branch,
    logisticsPlacement (array[string], optional): Logistics placement
}
MaterialV4 {
    itemNumber (string): Identifies the material,
    returnDate (string): Return date of the material if it is lendout. Format: yyyy-mm-dd,
    materialGroup (MaterialGroup): Name of the material group that the material belongs to,
    periodical (Periodical, optional): Present if material is a periodical,
    available (boolean): True if material is available on-shelf, false if lent out
}
MaterialGroup {
    name (string): Name of the material group,
    description (string, optional): Description of the material group
}
Periodical {
    volume (string, optional),
    volumeYear (string, optional),
    displayText (string): A representation of the periodica volume information that is suitable for display,
    volumeNumber (string, optional)
}
PlacementV1 {
    section (AgencySection, optional): Placement section,
    location (AgencyLocation, optional): Placement location,
    sublocation (AgencySublocation, optional): Placement sublocation,
    department (AgencyDepartment, optional): Placement department
}
AgencySection {
    sectionId (string): Section identifier,
    title (string): Name of the section
```

}
AgencyLocation {
  locationId (string): Location identifier,
  title (string): Name of the location
}
AgencySublocation {
  title (string): Name of the sub-location,
  sublocationId (string): Sub-location identifier
}
AgencyDepartment {
  departmentId (string): Department identifier,
  title (string): Name of the department
}
AgencyBranch {
  branchId (string): ISIL of branch (e.g. DK-761501),
  title (string): Name of branch
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/catalog/numberOfLoansPerRecord/v1 Get the number of loans for the given records in the given period.

Implementation Notes

Returns an array containing the amount of loans for each bibliographical record, created in the given period. If startDate and endDate is omitted, the number of loans for the given record is returned, without filtering on date. If a recordId is not found, it will not appear in the response array.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
recordid	Identifies the bibliographical records - The FAUST number.	query	array[string]
startDate	The start date of the time interval within which loans are counted for the specified records. Format: yyyy-mm-dd. If the parameter is not set, all loans until the endDate will be counted.	query	string
endDate	The end date of the time interval within which loans are counted for the specified records. Format: yyyy-mm-dd. if the parameter is not set, all loans since the startDate will be counted.	query	string

Response Class

Model | Model Schema

NumberOfLoansPerRecordV1 {
 recordId (string): The FAUST number of the Bibliographic record,
 numberOfLoans (integer): Number of loans on the record.
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

Interlibrary Loans

Show/Hide | List Operations | Expand Operations | Raw

POST /external/{agencyid}/ill/requestitem/v1 Initiate a request for an interlibrary loan for a given requesting agency, recordId, and enduserId.

Implementation Notes

The parameters for the requestingAgency, enduserId, and recordId are placed in the interlibrary loan request. If the record is a periodical, material periodical information also needs to be added in the request. The result of the request contains the field "successful", which can be true/false, and a result, which can be one of these values:

- PATRON_NOT_FOUND: No library loaner exists for the requesting agency
- RECORD_NOT_FOUND: No record exists for the given recordId
- MATERIAL_NOT_RESERVABLE: None of the materials on the record can be reserved
- SENDING_FAILURE: We were not able to send the item requested message to the requesting agency
- OK: request was successful

The request is only successful if the result is OK, and successful is set to true.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. NO-2030000)**	path	string
interlibraryLoanRequest	**interlibrary loan request**	body	Model | Model Schema **InterlibraryLoanRequestV1 {** **recordId** (string): Cicero-ID of the Bibliographic Record being requested, **periodical** (PeriodicalInformation, *optional*): Mandatory if requesting a periodical, **requestingAgency** (string): AgencyISIL for the requesting agency initiating the InterlibraryLoan, **enduserId** (string): Patron identifier number (e.g. library card number or social security card number) for the patron requesting the InterlibraryLoan **}** **PeriodicalInformation {** **volume** (string, *optional*), **number** (string, *optional*), **year** (string, *optional*) **}**

Response Class

Model | Model Schema

InterlibraryLoanResponseV1 {
  result (string) = ['PATRON_NOT_FOUND' or 'RECORD_NOT_FOUND' or 'MATERIAL_NOT_RESERVABLE' or 'SENDING_FAILURE' or 'OK'],
  successful (boolean): True if the operation was successful, False if not
}

Response Content Type application/json

Newsletter

Show/Hide | List Operations | Expand Operations | Raw

POST /external/{agencyid}/newsletters/preferences/patron/v1

Fetches the list of newsletters with preferences that a patron subscribed to with an email.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
subscriptionRequest	**patronId and email**	body	Model | Model Schema **NewsletterPatronSubscriptionRequestV1 {** **emailAddress** (string): The email address of the patron, **patronId** (integer): The id of the patron **}**

Response Class

Model | Model Schema

NewsletterV1 {
  listId (string): Newsletter contact list id,
  name (string): The newslatter name,
  tags (array[NewsletterTagV1], *optional*): The newsletter preferences,
  interestCategories (array[NewsletterGroupCategoryV1], *optional*): The newsletter preferences
}
NewsletterTagV1 {
  name (string): The preference name stored in Mailchimp,
  id (string): The preference id stored in Mailchimp

}
NewsletterGroupCategoryV1 {
  id (string): The preference id stored in Mailchimp,
  title (string): The preference name stored in Mailchimp,
  interests (array[NewsletterGroupNameV1], *optional*): The preference name stored in Mailchimp
}
NewsletterGroupNameV1 {
  name (string): The preference name stored in Mailchimp,
  id (string): The preference id stored in Mailchimp
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	Mailchimp configuration not found	

GET /external/{agencyid}/newsletters/preferences/v1

Fetches the newsletters with preferences from the Mailchimp platform configured on the Agency

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string

Response Class

Model | Model Schema

NewsletterV1 {
  listId (string): Newsletter contact list id,
  name (string): The newslatter name,

```
  tags (array[NewsletterTagV1], optional): The newsletter preferences,
  interestCategories (array[NewsletterGroupCategoryV1], optional): The newsletter preferences
}
NewsletterTagV1 {
  name (string): The preference name stored in Mailchimp,
  id (string): The preference id stored in Mailchimp
}
NewsletterGroupCategoryV1 {
  id (string): The preference id stored in Mailchimp,
  title (string): The preference name stored in Mailchimp,
  interests (array[NewsletterGroupNameV1], optional): The preference name stored in Mailchimp
}
NewsletterGroupNameV1 {
  name (string): The preference name stored in Mailchimp,
  id (string): The preference id stored in Mailchimp
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	Mailchimp configuration not found	

PUT /external/{agencyid}/newsletters/subscribe/v1

Subscribe to newsletter preferences with an email from the Mailchimp platform configured on the Agency

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string

Parameter	Description	Parameter Type	Data Type
`subscriptionDetails`		body	Model | Model Schema **NewsletterSubscriptionDetailsV1 {** **emailAddress** (string): Patron email, **subscriptions** (array[NewsletterV1], *optional*): Subscription details **}** **NewsletterV1 {** **listId** (string): Newsletter contact list id, **name** (string): The newslatter name, **tags** (array[NewsletterTagV1], *optional*): The newsletter preferences, **interestCategories** (array[NewsletterGroupCategoryV1], *optional*): The newsletter preferences **}** **NewsletterTagV1 {** **name** (string): The preference name stored in Mailchimp, **id** (string): The preference id stored in Mailchimp **}** **NewsletterGroupCategoryV1 {** **id** (string): The preference id stored in Mailchimp, **title** (string): The preference name stored in Mailchimp, **interests** (array[NewsletterGroupNameV1], *optional*): The preference name stored in Mailchimp **}** **NewsletterGroupNameV1 {** **name** (string): The preference name stored in Mailchimp, **id** (string): The preference id stored in Mailchimp **}**

Response Messages

HTTP Status Code	Reason	Response Model
`400`	bad request	
`401`	client unauthorized	
`404`	Mailchimp configuration not found	

PUT /external/{agencyid}/newsletters/unsubscribe/v1

Unsubscribe from newsletter preferences with an email from the Mailchimp platform configured on the Agency

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**ISIL of the agency (e.g. DK-761500)**	path	string
`subscriptionDetails`		body	Model Model Schema **NewsletterSubscriptionDetailsV1 {** **emailAddress** (string): Patron email, **subscriptions** (array[NewsletterV1], *optional*): Subscription details **}** **NewsletterV1 {** **listId** (string): Newsletter contact list id, **name** (string): The newslatter name, **tags** (array[NewsletterTagV1], *optional*): The newsletter preferences, **interestCategories** (array[NewsletterGroupCategoryV1], *optional*): The newsletter preferences **}** **NewsletterTagV1 {** **name** (string): The preference name stored in Mailchimp, **id** (string): The preference id stored in Mailchimp **}** **NewsletterGroupCategoryV1 {**

Parameter	Description	Parameter Type	Data Type
			id (string): The preference id stored in Mailchimp, **title** (string): The preference name stored in Mailchimp, **interests** (array[NewsletterGroupNameV1], *optional*): The preference name stored in Mailchimp **}** **NewsletterGroupNameV1 {** **name** (string): The preference name stored in Mailchimp, **id** (string): The preference id stored in Mailchimp **}**

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	Mailchimp configuration not found	

Payment

Show/Hide | List Operations | Expand Operations | Raw

GET /external/{agencyid}/patron/{patronid}/fee/{feeId}/transactions/v1

List of transactions in Cicero for the fee with all available information about the transaction.

Implementation Notes

Returns array of transactions.

Each transactions in the response includes a 'type', which is used to distinguish between different types of transactions.

The list of available types currently is
creation
reduction
cancellation
payment
payment_reduction
reimbursement

write_off
write_off_reduction
While the type can be used by client systems to look up a suitable display message for the end user, it is important that unrecognized types are treated as 'other'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron that owns the fee with transactions**	path	integer
feeId	**the fee that the transactions belong to**	path	integer

Response Class

Model | Model Schema

FeeTransaction {
amount (number): The amount of the transaction, in the currency of the agency,
creationTime (string): The date and time the transaction was created,
type (string): Can be used to distinguish between different types of transactions,
feeId (integer): Identifies the fee,
transactionId (integer): Identifies the transaction,
parentTransactionId (integer, *optional*): The id of the parent transaction if it has one.
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patron/{patronid}/fee/expected-fees/v1

Returns a list of expected fees of a patron, if all overdue loans were to be returned today.

Implementation Notes

This endpoint is intended for use with event-based fees.

If Cicero is configured to use time-based fees, this operation will always return an empty list.

Note: Expected fees vary from normal fees in the following ways:

The fee id (bill number) is -1.
The creation date is today.
The due date is null.
The payableByClient field is false.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron that owns the fees**	path	integer

Response Class

Model | Model Schema

FeeV2 {

payableByClient (boolean): true if the client system is allowed to offer payment for the fee, false if not allowed,
amount (number): The amount to pay, in the currency of the agency,
paidDate (string, *optional*): If the fee has been paid in full, this will be set to the date of the final payment, otherwise not set,
materials (array[FeeMaterialV2]): Set if fee covers materials,
reasonMessage (string): Human readable free text message about the reason for the fee, presentable to an end user (language is likely to be the mother tongue of the agency),
dueDate (string, *optional*): Expected payment due date,
type (string): Can be used to distinguish between different types of fees,
creationDate (string): The date the fee was created,
feeId (integer): Identifies the fee, used when registering a payment that covers the fee

}

FeeMaterialV2 {

recordId (string): The FAUST number of the bibliographic record,

materialGroup (MaterialGroup): The material group that the material belongs to,
periodical (Periodical, *optional*): Present if material is a periodical,
materialItemNumber (string): Identifies the exact material covered by the fee
}
MaterialGroup {
name (string): Name of the material group,
description (string, *optional*): Description of the material group
}
Periodical {
volume (string, *optional*),
volumeYear (string, *optional*),
displayText (string): A representation of the periodica volume information that is suitable for display,
volumeNumber (string, *optional*)
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patron/{patronid}/fees/v2 List of fees in FBS for the patron with all available information about the fee.

Implementation Notes

Returns array of fees.

If the fee covers loaned materials, information about the materials is returned. Each fee in the response includes a 'type', which is used to distinguish between different types of fees.

If the material exists no more, which is the case for fees that are related to closed interlibraryloans, then the fee is still returned, but without material information

The list of available types currently is
fee

compensation
While the type can be used by client systems to look up a suitable display message for the end user, it is important that unrecognized types are treated as 'other'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron that owns the fees**	path	integer
includepaid	true if all paid/unpaid fees should be included, false if only unpaid fees should be included; default=false	query	boolean
includenonpayable	true if fees that are not payable through a CMS system should be included (for read only access); default=false	query	boolean

Response Class

Model | Model Schema

FeeV2 {
payableByClient (boolean): true if the client system is allowed to offer payment for the fee, false if not allowed,
amount (number): The amount to pay, in the currency of the agency,
paidDate (string, *optional*): If the fee has been paid in full, this will be set to the date of the final payment, otherwise not set,
materials (array[FeeMaterialV2]): Set if fee covers materials,
reasonMessage (string): Human readable free text message about the reason for the fee, presentable to an end user (language is likely to be the mother tongue of the agency),
dueDate (string, *optional*): Expected payment due date,
type (string): Can be used to distinguish between different types of fees,
creationDate (string): The date the fee was created,
feeId (integer): Identifies the fee, used when registering a payment that covers the fee
}
FeeMaterialV2 {
recordId (string): The FAUST number of the bibliographic record,
materialGroup (MaterialGroup): The material group that the material belongs to,
periodical (Periodical, *optional*): Present if material is a periodical,

```
  materialItemNumber (string): Identifies the exact material covered by the fee
}
MaterialGroup {
  name (string): Name of the material group,
  description (string, optional): Description of the material group
}
Periodical {
  volume (string, optional),
  volumeYear (string, optional),
  displayText (string): A representation of the periodica volume information that is suitable for display,
  volumeNumber (string, optional)
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patron/{patronid}/payment/v2

Pay fees.

Implementation Notes

Returns array of payment confirmations for each fee.

This call is used to inform FBS that a payment have been completed successful from the payment gateway through the CMS client system. The payment contain the order ID from the payment gateway (e.g. dibs) and the fee identifiers for fees covered by the payment. It is expected that a fee has been paid in full when covered by a payment order. The client system is not allowed to offer partial payment of individual fees.

The paymentStatus on the response can be any of these values:
- paymentRegistered
- paymentRegisteredByDifferentOrderId
- paymentNotAllowedByClient
If any other value is encountered, it should be treated as yet another reason for not registerering payment of a fee using the specified order id.

Multiple calls to pay a fee with the same order Id will return the same confirmationId, and the payment will have paymentStatus=='paymentRegistered'.

If a fee has already been paid using a different orderId then no confirmationId is provided, and the payment will have paymentStatus=='paymentRegisteredByDifferentOrderId'.

If the client system was not allowed to offer payment of a fee, then no confirmationId is provided, and the payment will have paymentStatus=='paymentNotAllowedByClient'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron that owns the fees**	path	integer
paymentOrder	**registration of fees covered by a payment order**	body	Model Model Schema **PaymentOrder {** **orderId** (string): Order Id from payment gateway, **feeIds** (array[integer]): Array of fees fully covered by the order **}**

Response Class

Model Model Schema

PaymentConfirmationV2 {
orderId (string): Order Id from payment gateway,
confirmationId (string, *optional*): set if fee was registered when using the orderId, unset otherwise (see paymentStatus for reason),
feeId (integer),
paymentStatus (string)
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

Material Loans

Show/Hide | List Operations | Expand Operations | Raw

GET /external/{agencyid}/patrons/{patronid}/loans/{bookingid}/v2 Retrieves material loans for the given booking ID.

Implementation Notes

Retrieves an array of BookingLoan corresponding to the given booking ID.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the ID of the patron that owns the bookings**	path	integer
bookingid	**the ID of the booking**	path	string

Response Class

Model | Model Schema

BookingLoanV2 {
patronName (string): The name of the patron that owns the booking,
isRenewable (boolean): indicates whether this loan can be renewed,
loanDetails (LoanDetailsV2): The loan that was attempted renewed,
isLongtermLoan (boolean): indicates whether this loan is a long term loan,
renewalStatusList (array[string]): if isRenewable == false then this states the reasons for denial
}
LoanDetailsV2 {
recordId (string): The FAUST number of the bibliographic record,
loanType (string): The loan type, either **loan** or **interLibraryLoan**,

```
  materialGroup (MaterialGroup): Material group that the material belongs to,
  periodical (Periodical, optional): Present if material is a periodical,
  dueDate (string): The date when the material must be returned,
  ilBibliographicRecord (ILLBibliographicRecord, optional): Additional bibliographic information for inter-library loans,
  loanDate (string): The date when the material was picked up,
  materialItemNumber (string): Identifies the exact material that has been loaned,
  loanId (integer): Identifies the loan for use when renewing the loan
}
MaterialGroup {
  name (string): Name of the material group,
  description (string, optional): Description of the material group
}
Periodical {
  volume (string, optional),
  volumeYear (string, optional),
  displayText (string): A representation of the periodica volume information that is suitable for display,
  volumeNumber (string, optional)
}
ILLBibliographicRecord {
  author (string, optional): The author of the material,
  isbn (string, optional): ISBN-information from the bibliographic record,
  periodicalNumber (string, optional): Issue number of a periodical,
  edition (string, optional): Edition-information from the bibliographic record,
  language (string, optional): Language of the requested material.,
  bibliographicCategory (string, optional): Bibliographic category from danMARC2 008 *t,
  title (string, optional): The title of the material,
  publicationDateOfComponent (string, optional): Publication date of an item component, or article.,
  recordId (string): The FAUST number,
  issn (string, optional): ISSN-information from the bibliographic record,
  placeOfPublication (string, optional),
  mediumType (string): Type of the requested material - from danMARC2 009 *a+*g (general and specific),
  periodicalVolume (string, optional): Volume name of a periodical,
  publisher (string, optional): Publisher of the requested material.,
  publicationDate (string, optional): Publication date of the requested material.
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request com.dantek.dl.rest.RestException	
401	client unauthorized	
404	patron not found	

POST /external/{agencyid}/patrons/{patronid}/loans/renew/v2 Renew loans.

Implementation Notes

Returns an array of the updated loans.

If the materials could not be renewed, the return date will be unchanged.

The response field renewalStatus will contain a list of one or more of these values:

- renewed
- deniedReserved
- deniedMaxRenewalsReached
- deniedLoanerIsBlocked
- deniedMaterialIsNotLoanable
- deniedMaterialIsNotFound
- deniedLoanerNotFound
- deniedLoaningProfileNotFound
- deniedOtherReason

If any other value is encountered then it must be treated as 'deniedOtherReason'.

The response contains the field loanDetails.loanType, which can be any of these values:
- loan
- interLibraryLoan

The values are subject to change. If an unrecognized value is encountered, it should be treated as 'other' .

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string

Parameter	Description	Parameter Type	Data Type
`patronid`	**the patron that owns the loans**	path	integer
`materialLoanIds`	**a list of loanId to be renewed**	body	array[integer]

Response Class

Model | Model Schema

RenewedLoanV2 {
loanDetails (LoanDetailsV2): The loan that was attempted renewed,
renewalStatus (array[string]): indicates if renewal was succesful or denied - including the reason for denial.
}
LoanDetailsV2 {
recordId (string): The FAUST number of the bibliographic record,
loanType (string): The loan type, either **loan** or **interLibraryLoan**,
materialGroup (MaterialGroup): Material group that the material belongs to,
periodical (Periodical, *optional*): Present if material is a periodical,
dueDate (string): The date when the material must be returned,
ilBibliographicRecord (ILLBibliographicRecord, *optional*): Additional bibliographic information for inter-library loans,
loanDate (string): The date when the material was picked up,
materialItemNumber (string): Identifies the exact material that has been loaned,
loanId (integer): Identifies the loan for use when renewing the loan
}
MaterialGroup {
name (string): Name of the material group,
description (string, *optional*): Description of the material group
}
Periodical {
volume (string, *optional*),
volumeYear (string, *optional*),
displayText (string): A representation of the periodica volume information that is suitable for display,
volumeNumber (string, *optional*)
}
ILLBibliographicRecord {
author (string, *optional*): The author of the material,
isbn (string, *optional*): ISBN-information from the bibliographic record,

periodicalNumber (string, *optional*): Issue number of a periodical,
edition (string, *optional*): Edition-information from the bibliographic record,
language (string, *optional*): Language of the requested material.,
bibliographicCategory (string, *optional*): Bibliographic category from danMARC2 008 *t,
title (string, *optional*): The title of the material,
publicationDateOfComponent (string, *optional*): Publication date of an item component, or article.,
recordId (string): The FAUST number,
issn (string, *optional*): ISSN-information from the bibliographic record,
placeOfPublication (string, *optional*),
mediumType (string): Type of the requested material - from danMARC2 009 *a+*g (general and specific),
periodicalVolume (string, *optional*): Volume name of a periodical,
publisher (string, *optional*): Publisher of the requested material.,
publicationDate (string, *optional*): Publication date of the requested material.
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	patron not found	

GET /external/{agencyid}/patrons/{patronid}/loans/v2

Get list of current loans by the patron.

Implementation Notes

Returns an array of loans.

If a loan is not renewable then the field renewalStatus will contain a list of one or more of these values:
- deniedReserved
- deniedMaxRenewalsReached
- deniedLoanerIsBlocked
- deniedMaterialIsNotLoanable
- deniedMaterialIsNotFound
- deniedLoanerNotFound

- deniedLoaningProfileNotFound
- deniedOtherReason

If any other value is encountered then it must be treated as 'deniedOtherReason'.

The response contains the field loanDetails.loanType, which can be any of these values:
- loan
- interLibraryLoan

The values are subject to change. If an unrecognized value is encountered, it should be treated as 'other' .

NOTE: Cicero can decide to skip evaluation of the returned loans to minimize response time for loaners with many loans. In that case isRenewable will have the value true, as if it were a successful validation.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron that owns the loans**	path	integer

Response Class

Model | Model Schema

LoanV2 {
isRenewable (boolean): indicates whether this loan can be renewed,
loanDetails (LoanDetailsV2): The loan that was attempted renewed,
isLongtermLoan (boolean): indicates whether this loan is a long term loan,
renewalStatusList (array[string]): if isRenewable == false then this states the reasons for denial
}

LoanDetailsV2 {
recordId (string): The FAUST number of the bibliographic record,
loanType (string): The loan type, either **loan** or **interLibraryLoan**,
materialGroup (MaterialGroup): Material group that the material belongs to,
periodical (Periodical, *optional*): Present if material is a periodical,
dueDate (string): The date when the material must be returned,
ilBibliographicRecord (ILLBibliographicRecord, *optional*): Additional bibliographic information for inter-library loans,
loanDate (string): The date when the material was picked up,
materialItemNumber (string): Identifies the exact material that has been loaned,
loanId (integer): Identifies the loan for use when renewing the loan

}

MaterialGroup {
  name (string): Name of the material group,
  description (string, *optional*): Description of the material group
}

Periodical {
  volume (string, *optional*),
  volumeYear (string, *optional*),
  displayText (string): A representation of the periodica volume information that is suitable for display,
  volumeNumber (string, *optional*)
}

ILLBibliographicRecord {
  author (string, *optional*): The author of the material,
  isbn (string, *optional*): ISBN-information from the bibliographic record,
  periodicalNumber (string, *optional*): Issue number of a periodical,
  edition (string, *optional*): Edition-information from the bibliographic record,
  language (string, *optional*): Language of the requested material.,
  bibliographicCategory (string, *optional*): Bibliographic category from danMARC2 008 *t,
  title (string, *optional*): The title of the material,
  publicationDateOfComponent (string, *optional*): Publication date of an item component, or article.,
  recordId (string): The FAUST number,
  issn (string, *optional*): ISSN-information from the bibliographic record,
  placeOfPublication (string, *optional*),
  mediumType (string): Type of the requested material - from danMARC2 009 *a+*g (general and specific),
  periodicalVolume (string, *optional*): Volume name of a periodical,
  publisher (string, *optional*): Publisher of the requested material.,
  publicationDate (string, *optional*): Publication date of the requested material.
}

Response Content Type: application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	

HTTP Status Code	Reason	Response Model
401	client unauthorized	
404	patron not found	

GET /external/{agencyid}/patrons/{patronid}/loans/withhistoricalloans/v1 Retrieves material loans and historical loans for the given patron.

Implementation Notes

Retrieves an object containg two arrays of loans and historical loans corresponding to the given patron ID. If the patron has not given consent to keep historical data, no historical loans will be retrieved.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the ID of the patron that owns the loans**	path	integer

Response Class

Model | Model Schema

LoansWithHistoryV1 {
loans (array[LoanV2]): List of loans,
historicalMaterialLoans (array[HistoricalLoanV1]): List of historical material loans
}
LoanV2 {
isRenewable (boolean): indicates whether this loan can be renewed,
loanDetails (LoanDetailsV2): The loan that was attempted renewed,
isLongtermLoan (boolean): indicates whether this loan is a long term loan,
renewalStatusList (array[string]): if isRenewable == false then this states the reasons for denial
}
LoanDetailsV2 {
recordId (string): The FAUST number of the bibliographic record,
loanType (string): The loan type, either **loan** or **interLibraryLoan**,

```
  materialGroup (MaterialGroup): Material group that the material belongs to,
  periodical (Periodical, optional): Present if material is a periodical,
  dueDate (string): The date when the material must be returned,
  ilBibliographicRecord (ILLBibliographicRecord, optional): Additional bibliographic information for inter-library loans,
  loanDate (string): The date when the material was picked up,
  materialItemNumber (string): Identifies the exact material that has been loaned,
  loanId (integer): Identifies the loan for use when renewing the loan
}
MaterialGroup {
  name (string): Name of the material group,
  description (string, optional): Description of the material group
}
Periodical {
  volume (string, optional),
  volumeYear (string, optional),
  displayText (string): A representation of the periodica volume information that is suitable for display,
  volumeNumber (string, optional)
}
ILLBibliographicRecord {
  author (string, optional): The author of the material,
  isbn (string, optional): ISBN-information from the bibliographic record,
  periodicalNumber (string, optional): Issue number of a periodical,
  edition (string, optional): Edition-information from the bibliographic record,
  language (string, optional): Language of the requested material.,
  bibliographicCategory (string, optional): Bibliographic category from danMARC2 008 *t,
  title (string, optional): The title of the material,
  publicationDateOfComponent (string, optional): Publication date of an item component, or article.,
  recordId (string): The FAUST number,
  issn (string, optional): ISSN-information from the bibliographic record,
  placeOfPublication (string, optional),
  mediumType (string): Type of the requested material - from danMARC2 009 *a+*g (general and specific),
  periodicalVolume (string, optional): Volume name of a periodical,
  publisher (string, optional): Publisher of the requested material.,
  publicationDate (string, optional): Publication date of the requested material.
}
HistoricalLoanV1 {
```

recordId (string): The FAUST number of the bibliographic record,
loanType (string): The loan type, either **loan** or **interLibraryLoan**,
periodical (Periodical, *optional*): Present if material is a periodical,
dueDate (string): The date when the material must be returned,
ilBibliographicRecord (ILLBibliographicRecord, *optional*): Additional bibliographic information for inter-library loans,
loanDate (string): The date when the material was picked up,
materialItemNumber (string): Identifies the exact material that has been loaned,
loanId (integer): Identifies the loan for use when renewing the loan,
returnedDate (string, *optional*): Date when the material was returned
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request com.dantek.dl.rest.RestException	
401	client unauthorized	
404	patron not found	

Reservations version 2

Show/Hide | List Operations | Expand Operations | Raw

PUT /external/{agencyid}/patrons/{patronid}/reservations/v2 Update existing reservations.

Implementation Notes

Returns an array of the updated reservation details.

The response contains reservation state, which can be any of these values:
- reserved
- readyForPickup
- interLibraryReservation
- inTransit
- other

The values are subject to change. If an unrecognized value is encountered, it should be treated as 'other'.

The response contains loanType, which can be any of these values:

- loan
- interLibraryLoan

The values are subject to change. If an unrecognized value is encountered, it should be treated as 'other' .

The activation date can only be updated for active or passive reservations

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**ISIL of the agency (e.g. DK-761500)**	path	string
`patronid`	**the patron that owns the reservations**	path	integer
`reservations`	**the reservations to be updated**	body	Model | Model Schema **UpdateReservationBatchV2 {** **reservations** (array[UpdateReservationV2]) **}** **UpdateReservationV2 {** **expiryDate** (string, *optional*): The date where the patron is no longer interested in the reserved material. If not set, the reservation will keep the original date., **pickupBranch** (string, *optional*): ISIL-number of pickup branch. If not set, the reservation will keep the original pickup branch., **reservationId** (integer): Identifies the reservation, **activationDate** (string, *optional*): Sets the activation date. From this date, the reservation can be fulfilled. **}**

Response Class

Model | Model Schema

ResservationDetailsV3 {

pickupBranch (string): ISIL-number of pickup branch,
pickupDeadline (string, *optional*): Set if reserved material is available for loan,
dateOfReservation (string),
ilBibliographicRecord (ILLBibliographicRecord, *optional*): Additional bibliographic information for inter-library loans,
numberInQueue (integer, *optional*): The number in the reservation queue.,

pickupNumber (string, *optional*): The reservation number. Will be present if the reservation is ready for pickup (the state is 'readyForPickup'),
transactionId (string): Identifies the transaction of reservations,
recordId (string): The FAUST number,
expiryDate (string): The date when the patron is no longer interested in the reserved material,
reservationId (integer): Identifies the reservation for use when updating or deleting the reservation,
periodical (Periodical, *optional*): Present if material is a periodical,
reservationType (string),
state (string),
activationDate (string, *optional*): Sets the activation date. From this date, the reservation can be fulfilled.
}
ILLBibliographicRecord {
author (string, *optional*): The author of the material,
isbn (string, *optional*): ISBN-information from the bibliographic record,
periodicalNumber (string, *optional*): Issue number of a periodical,
edition (string, *optional*): Edition-information from the bibliographic record,
language (string, *optional*): Language of the requested material.,
bibliographicCategory (string, *optional*): Bibliographic category from danMARC2 008 *t,
title (string, *optional*): The title of the material,
publicationDateOfComponent (string, *optional*): Publication date of an item component, or article.,
recordId (string): The FAUST number,
issn (string, *optional*): ISSN-information from the bibliographic record,
placeOfPublication (string, *optional*),
mediumType (string): Type of the requested material - from danMARC2 009 *a+*g (general and specific),
periodicalVolume (string, *optional*): Volume name of a periodical,
publisher (string, *optional*): Publisher of the requested material.,
publicationDate (string, *optional*): Publication date of the requested material.
}
Periodical {
volume (string, *optional*),
volumeYear (string, *optional*),
displayText (string): A representation of the periodica volume information that is suitable for display,
volumeNumber (string, *optional*)
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/{patronid}/reservations/v3 — Get all unfulfilled reservations made by the patron.

Implementation Notes

Returns an array of reservation details.

When the patron picks up the reserved materials, the reservation will no longer be returned. Expired or deleted reservations will not be returned.

The response contains reservation state, which can be any of these values:
- reserved
- readyForPickup
- interLibraryReservation
- inTransit
- other

The values are subject to change. If an unrecognized value is encountered, it should be treated as 'other' .

The response contains reservationType, which can be any of these values:
- NORMAL
- PARALLEL
- SERIAL
- INTER_LIBRARY

The values are subject to change. If an unrecognized value is encountered, iit should be treated as 'normal'

The response contains a transactionId, which links together parallel reservations.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron that owns the reservations**	path	integer

Response Class

Model | Model Schema

ReservationDetailsV3 {
pickupBranch (string): ISIL-number of pickup branch,
pickupDeadline (string, *optional*): Set if reserved material is available for loan,
dateOfReservation (string),
ilBibliographicRecord (ILLBibliographicRecord, *optional*): Additional bibliographic information for inter-library loans,
numberInQueue (integer, *optional*): The number in the reservation queue.,
pickupNumber (string, *optional*): The reservation number. Will be present if the reservation is ready for pickup (the state is 'readyForPickup'),
transactionId (string): Identifies the transaction of reservations,
recordId (string): The FAUST number,
expiryDate (string): The date when the patron is no longer interested in the reserved material,
reservationId (integer): Identifies the reservation for use when updating or deleting the reservation,
periodical (Periodical, *optional*): Present if material is a periodical,
reservationType (string),
state (string),
activationDate (string, *optional*): Sets the activation date. From this date, the reservation can be fulfilled.
}
ILLBibliographicRecord {
author (string, *optional*): The author of the material,
isbn (string, *optional*): ISBN-information from the bibliographic record,
periodicalNumber (string, *optional*): Issue number of a periodical,
edition (string, *optional*): Edition-information from the bibliographic record,
language (string, *optional*): Language of the requested material.,
bibliographicCategory (string, *optional*): Bibliographic category from danMARC2 008 *t,
title (string, *optional*): The title of the material,
publicationDateOfComponent (string, *optional*): Publication date of an item component, or article.,
recordId (string): The FAUST number,
issn (string, *optional*): ISSN-information from the bibliographic record,
placeOfPublication (string, *optional*),
mediumType (string): Type of the requested material - from danMARC2 009 *a+*g (general and specific),
periodicalVolume (string, *optional*): Volume name of a periodical,
publisher (string, *optional*): Publisher of the requested material.,
publicationDate (string, *optional*): Publication date of the requested material.

}

Periodical {
  volume (string, *optional*),
  volumeYear (string, *optional*),
  displayText (string): A representation of the periodica volume information that is suitable for display,
  volumeNumber (string, *optional*)
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/{patronid}/reservations/v3 Create new reservations for the patron.

Implementation Notes

Given a CreateReservationBatch, it creates a list of reservations and returns a ReservationResponse.

The CreateReservationBatch.type indicates the reservation type of the request. If left out the request will be considered of type normal. The type can be any of the following values:

- normal
- parallel

The values are subject to change.

ReservationResponse.success indicates if the reservations were created sucessfully. If all reservations are successfully created ReservationResponse.success will be true. If the type is normal, and any of the reservations have failed then ReservationResponse.success will be false, and no reservations are created. If the type is parallel, and any of the reservations have failed then ReservationResponse.success will be false, but successful reservations are created.

ReservationResponse.reservationResults contains details about each reservation. A ReservationResult.result has the status of a reservation and can be any of the following values:

- success
- patron_is_blocked
- patron_not_found
- already_reserved
- already_loaned
- material_not_loanable
- material_not_reservable
- material_lost
- material_Discarded
- loaning_profile_not_found
- material_not_found
- material_part_of_collection
- not_reservable
- no_reservable_materials
- interlibrary_material_not_reservable
- previously_loaned_by_homebound_patron
- exceeds_max_reservations

The values are subject to change. If an unrecognized value is encountered, it should be treated as an error.

The reservation detail in the response contains a reservation state, which can be any of these values:
- reserved
- readyForPickup
- interLibraryReservation
- inTransit
- other

The values are subject to change. If an unrecognized value is encountered, it should be treated as 'other'.

When making a reservation of a periodical, the values to put in the PeriodicalReservation structure can be obtained from the periodical information retrieved with the Catalog service.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron that makes the reservations**	path	integer
createReservationBatch	**the reservations to be created**	body	Model | Model Schema **CreateReservationBatchV3 {** **reservations** (array[CreateReservationV2]): Reservations with duplicate record id's will be removed.,

Parameter	Description	Parameter Type	Data Type
			type (string, *optional*): Will be considered normal if not set } **CreateReservationV2 {** **recordId** (string): Identifies the bibliographical record to reserve - The FAUST number, **expiryDate** (string, *optional*): The date where the patron is no longer interested in the reserved material. If not set, a date will be calculated from the agency default interest period, **pickupBranch** (string, *optional*): ISIL-number of pickup branch. If not set, will default to patrons preferred pickup branch, **periodical** (PeriodicalReservation, *optional*): Present if making reservation on a periodical, **activationDate** (string, *optional*): Sets the activation date. From this date, the reservation can be fulfilled. **}** **PeriodicalReservation {** **volume** (string, *optional*), **volumeYear** (string, *optional*), **volumeNumber** (string, *optional*) **}**

Response Class

Model | Model Schema

ReservationResponseV3 {
success (boolean): True if all reservations were created successfully otherwise false,
reservationResults (array[ReservationResultV3]): Result of each reservation
}
ReservationResultV3 {
recordId (string): Recordid of the record to reserve,
result (string): The reservation result,
periodical (PeriodicalReservation, *optional*): Periodical information of the reservation,
reservationDetails (ReservationDetailsV3, *optional*): The reservation data as returned by the create/update operation

```
}
PeriodicalReservation {
   volume (string, optional),
   volumeYear (string, optional),
   volumeNumber (string, optional)
}
ReservationDetailsV3 {
   pickupBranch (string): ISIL-number of pickup branch,
   pickupDeadline (string, optional): Set if reserved material is available for loan,
   dateOfReservation (string),
   ilBibliographicRecord (ILLBibliographicRecord, optional): Additional bibliographic information for inter-library loans,
   numberInQueue (integer, optional): The number in the reservation queue.,
   pickupNumber (string, optional): The reservation number. Will be present if the reservation is ready for pickup (the state is 'readyForPickup'),
   transactionId (string): Identifies the transaction of reservations,
   recordId (string): The FAUST number,
   expiryDate (string): The date when the patron is no longer interested in the reserved material,
   reservationId (integer): Identifies the reservation for use when updating or deleting the reservation,
   periodical (Periodical, optional): Present if material is a periodical,
   reservationType (string),
   state (string),
   activationDate (string, optional): Sets the activation date. From this date, the reservation can be fulfilled.
}
ILLBibliographicRecord {
   author (string, optional): The author of the material,
   isbn (string, optional): ISBN-information from the bibliographic record,
   periodicalNumber (string, optional): Issue number of a periodical,
   edition (string, optional): Edition-information from the bibliographic record,
   language (string, optional): Language of the requested material.,
   bibliographicCategory (string, optional): Bibliographic category from danMARC2 008 *t,
   title (string, optional): The title of the material,
   publicationDateOfComponent (string, optional): Publication date of an item component, or article.,
   recordId (string): The FAUST number,
   issn (string, optional): ISSN-information from the bibliographic record,
   placeOfPublication (string, optional),
   mediumType (string): Type of the requested material - from danMARC2 009 *a+*g (general and specific),
   periodicalVolume (string, optional): Volume name of a periodical,
```

publisher (string, *optional*): Publisher of the requested material.,
publicationDate (string, *optional*): Publication date of the requested material.
}
Periodical {
volume (string, *optional*),
volumeYear (string, *optional*),
displayText (string): A representation of the periodica volume information that is suitable for display,
volumeNumber (string, *optional*)
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	patron not found	

Patron

Show/Hide | List Operations | Expand Operations | Raw

DELETE /external/{agencyid}/patrons/{patronid} Deletes a patron.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**Id of the patron to be deleted**	path	integer

POST /external/{agencyid}/patrons/{patronid}/addMailIdentifier/v1 Add the email as a LoanerIdentifier of type LibraryCard for the patron.

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**ISIL of the agency (e.g. DK-761500)**	path	string
`patronid`	**the patron concerned**	path	integer
`emailAddress`	**emailAddress belonging to the patron**	body	string

Response Class

Model | Model Schema

EmailIdentifierV1 {
result (string) = ['TOO_MANY_LIBRARY_CARDS_ALREADY' or 'EMAIL_ADDRESS_NOT_VERIFIED' or 'EMAIL_ADDRESS_DOES_NOT_BELONG_TO_LOANER' or 'IDENTIFIER_EXISTS_ON_AGENCY' or 'UNKNOWN_ERROR' or 'EMAIL_IDENTIFIER_CREATED']: result of the attempt at creating a email identifier for the patron,
success (boolean): true, if creation of email identifier was successful, false otherwise
}

Response Content Type application/json

GET /external/{agencyid}/patrons/{patronid}/favorites/v1 Retrieves a list of the patron's favorites lists

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**agencyId ISIL of the agency (e.g. DK-761500)**	path	string
`patronid`		path	integer

Response Class

Model | Model Schema

FavoritesV1 {
favorites (array[FavoriteV1]),
patronId (integer),
name (string)

```
}
FavoriteV1 {
  bibliographicRecordId (string),
  periodicalNumber (string, optional),
  periodicalVolume (string, optional),
  periodicalYear (string, optional)
}
```

Response Content Type application/json

PUT /external/{agencyid}/patrons/{patronid}/favorites/v1 Replaces the existing lists of favorites of the patron with the supplied lists.

Implementation Notes

The patron's favorites lists can be deleted by supplying an empty list.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**agencyId ISIL of the agency (e.g. DK-761500)**	path	string
patronid		path	integer
requestV1		body	array[UpdateFavoritesRequestV1]

PUT /external/{agencyid}/patrons/{patronid}/guardian/v1 Update information about the patron as a guardian.

Implementation Notes

If the corresponding agency configuration through /external/v1/{agencyid}/configuration/loaner.guardianVisibility.enabled is not enabled or there is no patron found, that has the guardian visibility enabled, for the patronid parameter, then response message 403 will be sent back.

The name and address cannot be supplied by the client. If the CPR-Registry is not authorized to provide information about the patron, then the name and address will not be updated.
It is possible to either update just the pincode, update just some patron settings, or update both.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron to be updated**	path	integer
updatePatronAsGuardian	**updated information about the patron**	body	Model | Model Schema **UpdatePatronAsGuardianRequestV1 {** **patron** (PatronSettingsV5, *optional*): Set this if patron details are to be changed, **pincodeChange** (PincodeChange, *optional*): Set this if pincode is to be changed, **guardianPersonId** (string): Identifies the guardian by CPR **}** **PatronSettingsV5 {** **emailAddress** (string, *optional*): Required if patron should receive email notifications Existing email addresses are overwritten with this value If left empty existing email addresses are deleted, **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place If left empty default library language will be used, **notificationProtocols** (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,

Parameter	Description	Parameter Type	Data Type
			phoneNumber (string, *optional*): Required if patron should receive SMS notifications Existing phonenumbers are overwritten with this value If left empty existing phonenumbers are deleted, **preferredPickupBranch** (string): ISIL-number of preferred pickup branch, **onHold** (Period, *optional*): If not set then the patron is not on hold, **guardianVisibility** (boolean), **receiveEmail** (boolean), **receivePostalMail** (boolean), **interests** (array[string], *optional*): A list of interests of the patron., **receiveSms** (boolean) **}** **Period {** **from** (string, *optional*): Open-ended if not set, **to** (string, *optional*): Open-ended if not set **}** **PincodeChange {** **pincode** (string): The new pincode for the libraryCard, **libraryCardNumber** (string): Identifies the libraryCard for which the pincode is to be changed. This can be either a physical card or the CPR number, that is used as a libraryCard **}**

Response Class

Model | Model Schema

AuthenticatedPatronV8 {

patron (PatronV7, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

```
}
PatronV7 {
  birthday (string, optional),
  secondaryAddress (AddressV2, optional),
  preferredLanguage (string, optional): Language in which the patron prefers the communication with the library to take place,
  preferredPickupBranch (string): ISIL of preferred pickup branch,
  address (AddressV2, optional),
  onHold (Period, optional): If not set then the patron is not on hold,
  patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
  guardianVisibility (boolean),
  receiveEmail (boolean),
  blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
  receiveSms (boolean),
  tags (array[string], optional): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
  emailAddress (string, optional),
  notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
  phoneNumber (string, optional),
  name (string, optional),
  receivePostalMail (boolean),
  allowBookings (boolean, optional): True if the user is allowed to create bookings.,
  defaultInterestPeriod (integer): Length of default interest period in days,
  interests (array[InterestV1], optional),
  resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
  country (string): Country,
  city (string): City,
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string): Postal code
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
```

blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}
InterestV1 {
displayName (string): Display name of the interest,
name (string): Name of the interest
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
`400`	bad request	
`401`	client unauthorized	
`403`	the guardian does not have access to update the patron details	

PUT /external/{agencyid}/patrons/{patronid}/guardian/v2 Update information about the patron as a guardian.

Implementation Notes

If the corresponding agency configuration through /external/v1/{agencyid}/configuration/loaner.guardianVisibility.enabled is not enabled or there is no patron found, that has the guardian visibility enabled, for the patronid parameter, then response message 403 will be sent back.

The name and address cannot be supplied by the client. If the configured person registry is not authorized to provide information about the patron, then the name and address will not be updated.
It is possible to either update just the pincode, update just some patron settings, or update both.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Multiple email addresses and phone numbers can be stored for a patron.

If multiple email addresses are supplied having receiveNotification as true, then only one of them will be randomly stored as preferred and the rest will be stored as not preferred.

If multiple phone numbers are supplied having receiveNotification as true, then only one of them will be randomly stored as preferred and the rest will be stored as not preferred.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron to be updated**	path	integer
updatePatronAsGuardian	**updated information about the patron**	body	Model | Model Schema **UpdatePatronAsGuardianRequestV2 {** **patron** (PatronSettingsV6, *optional*): Set this if patron details are to be changed, **pincodeChange** (PincodeChange, *optional*): Set this if pincode is to be changed, **guardianPersonId** (string): Identifies the guardian by person identifier **}** **PatronSettingsV6 {** **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place If left empty default library language will be used, **notificationProtocols** (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included., **emailAddresses** (array[EmailAddressV1], *optional*): Existing email addresses are overwritten with these

Parameter	Description	Parameter Type	Data Type

values If left empty existing email addresses are deleted,
preferredPickupBranch (string): ISIL-number of preferred pickup branch,
onHold (Period, *optional*): If not set then the patron is not on hold,
guardianVisibility (boolean),
receivePostalMail (boolean),
interests (array[string], *optional*): A list of interests of the patron.,
phoneNumbers (array[PhoneNumberV1], *optional*): Existing phonenumbers are overwritten with these values If left empty existing phonenumbers are deleted
}
EmailAddressV1 {
emailAddress (string),
receiveNotification (boolean)
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
PhoneNumberV1 {
receiveNotification (boolean),
phoneNumber (string)
}
PincodeChange {
pincode (string): The new pincode for the libraryCard,
libraryCardNumber (string): Identifies the libraryCard for which the pincode is to be changed. This can be either a physical card or the CPR number, that is used as a libraryCard
}

Response Class

Model | Model Schema

AuthenticatedPatronV10 {

patron (PatronV9, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV9 {

birthday (string, *optional*),
secondaryAddress (AddressV3, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV3, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
guardianVisibility (boolean),
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
keepLoanHistoricalData (boolean, *optional*): Patron consent to keep historical loans,
receiveSms (boolean),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
hasNationalRegistrySynchronizationConsent (boolean),
emailAddress (string, *optional*),
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
interests (array[string], *optional*),
resident (boolean): True if the user is resident in the same municipality as the library

}

AddressV3 {

country (string): Country,

```
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code,
    district (string): Dsitrict,
    subDistrict (string): Subdistrict
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
403	the guardian does not have access to update the patron details	

PUT /external/{agencyid}/patrons/{patronid}/identifiers/block/v1 Block identifier for the given patron.

Implementation Notes

The identifier for the given patron is blocked

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**agencyId ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the id of a given patron**	path	integer

blockIdentifier	**Identifier to be blocked**	body	Model | Model Schema
			BlockIdentifierV1 {
			identifier (string)
			}

GET /external/{agencyid}/patrons/{patronid}/identifiers/v1 Retrieve all non-expired identifiers for the given patron.

Implementation Notes

The identifiers are sorted by IdentifierType.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**agencyId ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the id of a given patron**	path	integer

Response Class

Model | Model Schema

LoanerIdentifierV1 {
expiryDate (string, *optional*),
identifier (string),
description (string, *optional*),
identifierType (string),
status (string)
}

Response Content Type application/json

PUT /external/{agencyid}/patrons/{patronid}/updateconsent/historicaldata/v1 Updates the patron’s consent to keep historical loans.

Implementation Notes

Withdrawing consent will permanently delete existing historical loans.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer
consent		body	boolean

Response Class

Model | Model Schema

AuthenticatedPatronV9 {

patron (PatronV8, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV8 {

birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
guardianVisibility (boolean),
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
keepLoanHistoricalData (boolean, *optional*): Patron consent to keep historical loans,

```
    receiveSms (boolean),
    tags (array[string], optional): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
    hasNationalRegistrySynchronizationConsent (boolean),
    emailAddress (string, optional),
    notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
    phoneNumber (string, optional),
    name (string, optional),
    receivePostalMail (boolean),
    allowBookings (boolean, optional): True if the user is allowed to create bookings.,
    defaultInterestPeriod (integer): Length of default interest period in days,
    interests (array[string], optional),
    resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
```

Response Content Type application/json

PUT /external/{agencyid}/patrons/{patronid}/updateconsent/historicaldata/v2 Updates the patron’s consent to keep historical loans.

Implementation Notes

Withdrawing consent will permanently delete existing historical loans. Returns list of given consents for the patron.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

updateConsent	**payload containing true or false consent**	body	Model Model Schema
			UpdateConsentV1 {
			consent (boolean): If consent is given or not.
			}

Response Class

Model Model Schema

ConsentV1 {
consentType (string): If you encounter another type of consent please add it to this documentation.

ConsentTypes:
- 'KEEP_HISTORICAL_LOAN_DATA'
- 'SYNC_WITH_NATIONAL_REGISTRY'

,
consentDate (string): Date that consent was given.
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
200	ok	
400	bad request	

HTTP Status Code	Reason	Response Model
401	client unauthorized	
403	forbidden access to wrong agency	
404	no patron found for patronId	

PUT /external/{agencyid}/patrons/{patronid}/updateconsent/nationalregistry/v1

Updates the patron's consent to having their data synchronized with the National Patron Registry.

Implementation Notes

The data that is synchronized can be configured for the library.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer
consent		body	boolean

Response Class

Model | Model Schema

AuthenticatedPatronV9 {
patron (PatronV8, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}
PatronV8 {
birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,

```
  preferredPickupBranch (string): ISIL of preferred pickup branch,
  address (AddressV2, optional),
  onHold (Period, optional): If not set then the patron is not on hold,
  patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
  guardianVisibility (boolean),
  receiveEmail (boolean),
  blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
  keepLoanHistoricalData (boolean, optional): Patron consent to keep historical loans,
  receiveSms (boolean),
  tags (array[string], optional): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
  hasNationalRegistrySynchronizationConsent (boolean),
  emailAddress (string, optional),
  notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
  phoneNumber (string, optional),
  name (string, optional),
  receivePostalMail (boolean),
  allowBookings (boolean, optional): True if the user is allowed to create bookings.,
  defaultInterestPeriod (integer): Length of default interest period in days,
  interests (array[string], optional),
  resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
  country (string): Country,
  city (string): City,
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string): Postal code
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
```

}

Response Content Type application/json

PUT /external/{agencyid}/patrons/{patronid}/updateconsent/nationalregistry/v2

Updates the patron's consent to having their data synchronized with the National Patron Registry.

Implementation Notes

The data that is synchronized can be configured for the library. Returns list of given consents for the patron.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

updateConsent	**payload containing true or false consent**	body	Model Model Schema
			UpdateConsentV1 {
			consent (boolean): If consent is given or not.
			}

Response Class

Model Model Schema

ConsentV1 {
consentType (string): If you encounter another type of consent please add it to this documentation.

ConsentTypes:
- 'KEEP_HISTORICAL_LOAN_DATA'
- 'SYNC_WITH_NATIONAL_REGISTRY'

,
consentDate (string): Date that consent was given.
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
200	ok	
400	bad request	
401	client unauthorized	
403	forbidden access to wrong agency	
404	no patron found for patronId	

GET /external/{agencyid}/patrons/{patronid}/v1

Returns the patron details

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website
The codes are informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

AuthenticatedPatronV5 {

patron (PatronV4, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,

authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV4 {

birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library

}

AddressV2 {

country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code

}

Period {

from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set

}

BlockStatus {

blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/{patronid}/v2

Returns the patron details

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website
The codes are informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

AuthenticatedPatronV6 {
patron (PatronV5, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,

```
  authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
  - 'VALID': successfully authenticated
  - 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
  - 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts
}
PatronV5 {
  birthday (string, optional),
  secondaryAddress (AddressV2, optional),
  preferredLanguage (string, optional): Language in which the patron prefers the communication with the library to take place,
  preferredPickupBranch (string): ISIL of preferred pickup branch,
  address (AddressV2, optional),
  onHold (Period, optional): If not set then the patron is not on hold,
  patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
  receiveEmail (boolean),
  blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
  receiveSms (boolean),
  emailAddress (string, optional),
  notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
  phoneNumber (string, optional),
  name (string, optional),
  receivePostalMail (boolean),
  allowBookings (boolean, optional): True if the user is allowed to create bookings.,
  defaultInterestPeriod (integer): Length of default interest period in days,
  resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
  country (string): Country,
  city (string): City,
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string): Postal code
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
```

```
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/{patronid}/v3

Returns the patron details

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website
The codes are informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

```
AuthenticatedPatronV7 {
    patron (PatronV6, optional): Only available if patron exists in FBS and was succesfully authenticated.,
    authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
    - 'VALID': successfully authenticated
    - 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
    - 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts
}
PatronV6 {
    birthday (string, optional),
    secondaryAddress (AddressV2, optional),
    preferredLanguage (string, optional): Language in which the patron prefers the communication with the library to take place,
    preferredPickupBranch (string): ISIL of preferred pickup branch,
    address (AddressV2, optional),
    onHold (Period, optional): If not set then the patron is not on hold,
    patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
    receiveEmail (boolean),
    blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
    receiveSms (boolean),
    tags (array[string], optional): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
    emailAddress (string, optional),
    notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
    phoneNumber (string, optional),
    name (string, optional),
    receivePostalMail (boolean),
    allowBookings (boolean, optional): True if the user is allowed to create bookings.,
    defaultInterestPeriod (integer): Length of default interest period in days,
    resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

PUT /external/{agencyid}/patrons/{patronid}/v3 Update information about the patron.

Implementation Notes

The name and address cannot be supplied by the client. If the CPR-Registry is not authorized to provide information about the patron, then the name and address will not be updated.

It is possible to either update just the pincode, update just some patron settings, or update both.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron to be updated**	path	integer
updatePatron	**updated information about the patron**	body	Model Model Schema **UpdatePatronRequestV3 {** **patron** (PatronSettingsV3, *optional*): Set this if patron details are to be changed, **pincodeChange** (PincodeChange, *optional*): Set this if pincode is to be changed **}** **PatronSettingsV3 {** **emailAddress** (string, *optional*): Required if patron should receive email notifications Existing email addresses are overwritten with this value If left empty existing email addresses are deleted, **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place If left empty default library language will be used, **phoneNumber** (string, *optional*): Required if patron should receive SMS notifications Existing phonenumbers are overwritten with this value If left empty existing phonenumbers are deleted, **preferredPickupBranch** (string): ISIL-number of preferred pickup branch, **onHold** (Period, *optional*): If not set then the patron is not on hold, **receiveEmail** (boolean), **receivePostalMail** (boolean), **receiveSms** (boolean) **}** **Period {** **from** (string, *optional*): Open-ended if not set, **to** (string, *optional*): Open-ended if not set **}** **PincodeChange {**

Parameter	Description	Parameter Type	Data Type
			pincode (string): The new pincode for the libraryCard, **libraryCardNumber** (string): Identifies the libraryCard for which the pincode is to be changed. This can be either a physical card or the CPR number, that is used as a libraryCard **}**

Response Class

Model | Model Schema

AuthenticatedPatronV3 {
 authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,
 patron (PatronV3, *optional*): Only available if patron exists in FBS and was succesfully authenticated.
}
PatronV3 {
 birthday (string, *optional*),
 secondaryAddress (Address, *optional*),
 preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
 preferredPickupBranch (string): ISIL of preferred pickup branch,
 address (Address, *optional*),
 onHold (Period, *optional*): If not set then the patron is not on hold,
 patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
 receiveEmail (boolean),
 blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
 receiveSms (boolean),
 emailAddress (string, *optional*),
 phoneNumber (string, *optional*),
 name (string, *optional*),
 receivePostalMail (boolean),
 allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
 defaultInterestPeriod (integer): Length of default interest period in days,
 resident (boolean): True if the user is resident in the same municipality as the library
}
Address {

```
   country (string),
   city (string),
   street (string): Street and number,
   coName (string): c/o name,
   postalCode (string)
}
Period {
   from (string, optional): Open-ended if not set,
   to (string, optional): Open-ended if not set
}
BlockStatus {
   blockedReason (string): Reason code for block,
   blockedSince (string),
   message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	patron not found	

GET /external/{agencyid}/patrons/{patronid}/v4

Returns the patron details

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**ISIL of the agency (e.g. DK-761500)**	path	string
`patronid`	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

AuthenticatedPatronV8 {
patron (PatronV7, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}
PatronV7 {
birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
guardianVisibility (boolean),
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
emailAddress (string, *optional*),
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*),
name (string, *optional*),

receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
interests (array[InterestV1], *optional*),
resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
BlockStatus {
blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}
InterestV1 {
displayName (string): Display name of the interest,
name (string): Name of the interest
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

PUT /external/{agencyid}/patrons/{patronid}/v4 Update information about the patron.

Implementation Notes

The name and address cannot be supplied by the client. If the CPR-Registry is not authorized to provide information about the patron, then the name and address will not be updated.

It is possible to either update just the pincode, update just some patron settings, or update both.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron to be updated**	path	integer
updatePatron	**updated information about the patron**	body	Model Model Schema **UpdatePatronRequestV3 {** **patron** (PatronSettingsV3, *optional*): Set this if patron details are to be changed, **pincodeChange** (PincodeChange, *optional*): Set this if pincode is to be changed **}** **PatronSettingsV3 {** **emailAddress** (string, *optional*): Required if patron should receive email notifications Existing email addresses are overwritten with this value If left empty existing email addresses are deleted, **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to

Parameter	Description	Parameter Type	Data Type
			take place If left empty default library language will be used, **phoneNumber** (string, *optional*): Required if patron should receive SMS notifications Existing phonenumbers are overwritten with this value If left empty existing phonenumbers are deleted, **preferredPickupBranch** (string): ISIL-number of preferred pickup branch, **onHold** (Period, *optional*): If not set then the patron is not on hold, **receiveEmail** (boolean), **receivePostalMail** (boolean), **receiveSms** (boolean) **}** **Period {** **from** (string, *optional*): Open-ended if not set, **to** (string, *optional*): Open-ended if not set **}** **PincodeChange {** **pincode** (string): The new pincode for the libraryCard, **libraryCardNumber** (string): Identifies the libraryCard for which the pincode is to be changed. This can be either a physical card or the CPR number, that is used as a libraryCard **}**

Response Class

Model | Model Schema

AuthenticatedPatronV4 {

authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,

patron (PatronV4, *optional*): Only available if patron exists in FBS and was succesfully authenticated.

}

PatronV4 {

birthday (string, *optional*),

secondaryAddress (AddressV2, *optional*),

```
    preferredLanguage (string, optional): Language in which the patron prefers the communication with the library to take place,
    preferredPickupBranch (string): ISIL of preferred pickup branch,
    address (AddressV2, optional),
    onHold (Period, optional): If not set then the patron is not on hold,
    patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
    receiveEmail (boolean),
    blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
    receiveSms (boolean),
    emailAddress (string, optional),
    phoneNumber (string, optional),
    name (string, optional),
    receivePostalMail (boolean),
    allowBookings (boolean, optional): True if the user is allowed to create bookings.,
    defaultInterestPeriod (integer): Length of default interest period in days,
    resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	patron not found	

PUT /external/{agencyid}/patrons/{patronid}/v5 Update information about the patron.

Implementation Notes

The name and address cannot be supplied by the client. If the CPR-Registry is not authorized to provide information about the patron, then the name and address will not be updated.

It is possible to either update just the pincode, update just some patron settings, or update both.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron to be updated**	path	integer
updatePatron	**updated information about the patron**	body	Model | Model Schema **UpdatePatronRequestV4 {** **patron** (PatronSettingsV4, *optional*): Set this if patron details are to be changed, **pincodeChange** (PincodeChange, *optional*): Set this if pincode is to be changed

Parameter	Description	Parameter Type	Data Type
			} **PatronSettingsV4 {** **emailAddress** (string, *optional*): Required if patron should receive email notifications Existing email addresses are overwritten with this value If left empty existing email addresses are deleted, **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place If left empty default library language will be used, **notificationProtocols** (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included., **phoneNumber** (string, *optional*): Required if patron should receive SMS notifications Existing phonenumbers are overwritten with this value If left empty existing phonenumbers are deleted, **preferredPickupBranch** (string): ISIL-number of preferred pickup branch, **onHold** (Period, *optional*): If not set then the patron is not on hold, **receiveEmail** (boolean), **receivePostalMail** (boolean), **receiveSms** (boolean) } **Period {** **from** (string, *optional*): Open-ended if not set, **to** (string, *optional*): Open-ended if not set } **PincodeChange {** **pincode** (string): The new pincode for the libraryCard, **libraryCardNumber** (string): Identifies the libraryCard for which the pincode is to be changed. This can be either a physical card or the CPR number, that is used as a libraryCard }

Response Class

Model | Model Schema

AuthenticatedPatronV6 {
  patron (PatronV5, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
  authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
  - 'VALID': successfully authenticated
  - 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
  - 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV5 {
  birthday (string, *optional*),
  secondaryAddress (AddressV2, *optional*),
  preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
  preferredPickupBranch (string): ISIL of preferred pickup branch,
  address (AddressV2, *optional*),
  onHold (Period, *optional*): If not set then the patron is not on hold,
  patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
  receiveEmail (boolean),
  blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
  receiveSms (boolean),
  emailAddress (string, *optional*),
  notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
  phoneNumber (string, *optional*),
  name (string, *optional*),
  receivePostalMail (boolean),
  allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
  defaultInterestPeriod (integer): Length of default interest period in days,
  resident (boolean): True if the user is resident in the same municipality as the library

}

AddressV2 {
  country (string): Country,
  city (string): City,
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string): Postal code

}
Period {
  from (string, *optional*): Open-ended if not set,
  to (string, *optional*): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	patron not found	

PUT /external/{agencyid}/patrons/{patronid}/v6 Update information about the patron.

Implementation Notes

The name and address cannot be supplied by the client. If the CPR-Registry is not authorized to provide information about the patron, then the name and address will not be updated.

It is possible to either update just the pincode, update just some patron settings, or update both.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron to be updated**	path	integer
updatePatron	**updated information about the patron**	body	Model Model Schema **UpdatePatronRequestV4 {** **patron** (PatronSettingsV4, *optional*): Set this if patron details are to be changed, **pincodeChange** (PincodeChange, *optional*): Set this if pincode is to be changed **}** **PatronSettingsV4 {** **emailAddress** (string, *optional*): Required if patron should receive email notifications Existing email addresses are overwritten with this value If left empty existing email addresses are deleted, **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place If left empty default library language will be used, **notificationProtocols** (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included., **phoneNumber** (string, *optional*): Required if patron should receive SMS notifications Existing phonenumbers are overwritten with this value If left empty existing phonenumbers are deleted, **preferredPickupBranch** (string): ISIL-number of preferred pickup branch, **onHold** (Period, *optional*): If not set then the patron is not on hold, **receiveEmail** (boolean), **receivePostalMail** (boolean),

Parameter	Description	Parameter Type	Data Type

			receiveSms (boolean)
			}
			Period {
			from (string, *optional*): Open-ended if not set,
			to (string, *optional*): Open-ended if not set
			}
			PincodeChange {
			pincode (string): The new pincode for the libraryCard,
			libraryCardNumber (string): Identifies the libraryCard for which the pincode is to be changed. This can be either a physical card or the CPR number, that is used as a libraryCard
			}

Response Class

Model | Model Schema

AuthenticatedPatronV7 {

patron (PatronV6, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV6 {

birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,

emailAddress (string, *optional*),
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
BlockStatus {
blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	patron not found	

PUT /external/{agencyid}/patrons/{patronid}/v7 Update information about the patron.

Implementation Notes

The name and address cannot be supplied by the client. If the CPR-Registry is not authorized to provide information about the patron, then the name and address will not be updated.

It is possible to either update just the pincode, update just some patron settings, or update both.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron to be updated**	path	integer
updatePatron	**updated information about the patron**	body	Model Model Schema **UpdatePatronRequestV5 {** **patron** (PatronSettingsV5, *optional*): Set this if patron details are to be changed, **pincodeChange** (PincodeChange, *optional*): Set this if pincode is to be changed **}** **PatronSettingsV5 {** **emailAddress** (string, *optional*): Required if patron should receive email notifications Existing email addresses are overwritten with this value If left empty existing email addresses are deleted, **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to

Parameter	Description	Parameter Type	Data Type

take place If left empty default library language will be used,
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*): Required if patron should receive SMS notifications Existing phonenumbers are overwritten with this value If left empty existing phonenumbers are deleted,
preferredPickupBranch (string): ISIL-number of preferred pickup branch,
onHold (Period, *optional*): If not set then the patron is not on hold,
guardianVisibility (boolean),
receiveEmail (boolean),
receivePostalMail (boolean),
interests (array[string], *optional*): A list of interests of the patron.,
receiveSms (boolean)
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
PincodeChange {
pincode (string): The new pincode for the libraryCard,
libraryCardNumber (string): Identifies the libraryCard for which the pincode is to be changed. This can be either a physical card or the CPR number, that is used as a libraryCard
}

Response Class

Model | Model Schema

AuthenticatedPatronV8 {
patron (PatronV7, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,

```
    authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
    - 'VALID': successfully authenticated
    - 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
    - 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts
}
PatronV7 {
    birthday (string, optional),
    secondaryAddress (AddressV2, optional),
    preferredLanguage (string, optional): Language in which the patron prefers the communication with the library to take place,
    preferredPickupBranch (string): ISIL of preferred pickup branch,
    address (AddressV2, optional),
    onHold (Period, optional): If not set then the patron is not on hold,
    patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
    guardianVisibility (boolean),
    receiveEmail (boolean),
    blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
    receiveSms (boolean),
    tags (array[string], optional): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
    emailAddress (string, optional),
    notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
    phoneNumber (string, optional),
    name (string, optional),
    receivePostalMail (boolean),
    allowBookings (boolean, optional): True if the user is allowed to create bookings.,
    defaultInterestPeriod (integer): Length of default interest period in days,
    interests (array[InterestV1], optional),
    resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
InterestV1 {
  displayName (string): Display name of the interest,
  name (string): Name of the interest
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	patron not found	

PUT /external/{agencyid}/patrons/{patronid}/v8 Update information about the patron.

Implementation Notes

The name and address cannot be supplied by the client. If the configured person registry is not authorized to provide information about the patron, then the name and address will not be updated.

It is possible to either update just the pincode, update just some patron settings, or update both.

Multiple email addresses and phone numbers can be stored for a patron.

If multiple email addresses are supplied having receiveNotification as true, then only one of them will be randomly stored as preferred and the rest will be stored as not preferred.

If multiple phone numbers are supplied having receiveNotification as true, then only one of them will be randomly stored as preferred and the rest will be stored as not preferred.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron to be updated**	path	integer
updatePatron	**updated information about the patron**	body	Model | Model Schema **UpdatePatronRequestV6 {** **patron** (PatronSettingsV6, *optional*): Set this if patron details are to be changed, **pincodeChange** (PincodeChange, *optional*): Set this if pincode is to be changed **}** **PatronSettingsV6 {** **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place If left empty default library language will be used, **notificationProtocols** (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included., **emailAddresses** (array[EmailAddressV1], *optional*): Existing email addresses are overwritten with these values If left empty existing email addresses are deleted, **preferredPickupBranch** (string): ISIL-number of preferred pickup branch,

Parameter	Description	Parameter Type	Data Type

onHold (Period, *optional*): If not set then the patron is not on hold,
guardianVisibility (boolean),
receivePostalMail (boolean),
interests (array[string], *optional*): A list of interests of the patron.,
phoneNumbers (array[PhoneNumberV1], *optional*): Existing phonenumbers are overwritten with these values If left empty existing phonenumbers are deleted
}
EmailAddressV1 {
emailAddress (string),
receiveNotification (boolean)
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
PhoneNumberV1 {
receiveNotification (boolean),
phoneNumber (string)
}
PincodeChange {
pincode (string): The new pincode for the libraryCard,
libraryCardNumber (string): Identifies the libraryCard for which the pincode is to be changed. This can be either a physical card or the CPR number, that is used as a libraryCard
}

Response Class

Model | Model Schema

Response

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	patron not found	
204	no content	

POST /external/{agencyid}/patrons/authenticate/v3

Authenticates a patron and returns the patron details.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
authenticationRequest	**credentials for patron to be authenticated**	body	Model | Model Schema **AuthenticationRequest {** **pincode** (string): The pincode that belongs to the libraryCardNumber in plain text, **libraryCardNumber** (string): Identifies a libraryCard. This can be either a physical card or a CPR number that is used as a libraryCard

Parameter	Description	Parameter Type	Data Type
			}

Response Class

Model | Model Schema

AuthenticatedPatronV3 {
  authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,
  patron (PatronV3, *optional*): Only available if patron exists in FBS and was succesfully authenticated.
}
PatronV3 {
  birthday (string, *optional*),
  secondaryAddress (Address, *optional*),
  preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
  preferredPickupBranch (string): ISIL of preferred pickup branch,
  address (Address, *optional*),
  onHold (Period, *optional*): If not set then the patron is not on hold,
  patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
  receiveEmail (boolean),
  blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
  receiveSms (boolean),
  emailAddress (string, *optional*),
  phoneNumber (string, *optional*),
  name (string, *optional*),
  receivePostalMail (boolean),
  allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
  defaultInterestPeriod (integer): Length of default interest period in days,
  resident (boolean): True if the user is resident in the same municipality as the library
}
Address {
  country (string),
  city (string),
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string)

}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
BlockStatus {
blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/authenticate/v4 Authenticates a patron and returns the patron details.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
authenticationRequest	**credentials for patron to be authenticated**	body	Model Model Schema **AuthenticationRequest {** **pincode** (string): The pincode that belongs to the libraryCardNumber in plain text, **libraryCardNumber** (string): Identifies a libraryCard. This can be either a physical card or a CPR number that is used as a libraryCard **}**

Response Class

Model Model Schema

AuthenticatedPatronV4 {
authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,
patron (PatronV4, *optional*): Only available if patron exists in FBS and was succesfully authenticated.
}
PatronV4 {
birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,

defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
BlockStatus {
blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/authenticate/v5 Authenticates a patron and returns the patron details.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron.

If a patron is blocked the reason is available as a code:

- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
authenticationRequest	**credentials for patron to be authenticated**	body	Model | Model Schema **AuthenticationRequest {** **pincode** (string): The pincode that belongs to the libraryCardNumber in plain text, **libraryCardNumber** (string): Identifies a libraryCard. This can be either a physical card or a CPR number that is used as a libraryCard **}**

Response Class

Model | Model Schema

AuthenticatedPatronV5 {

patron (PatronV4, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,

authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:

- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV4 {

birthday (string, *optional*),

secondaryAddress (AddressV2, *optional*),

preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,

preferredPickupBranch (string): ISIL of preferred pickup branch,

address (AddressV2, *optional*),

```
  onHold (Period, optional): If not set then the patron is not on hold,
  patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
  receiveEmail (boolean),
  blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
  receiveSms (boolean),
  emailAddress (string, optional),
  phoneNumber (string, optional),
  name (string, optional),
  receivePostalMail (boolean),
  allowBookings (boolean, optional): True if the user is allowed to create bookings.,
  defaultInterestPeriod (integer): Length of default interest period in days,
  resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
  country (string): Country,
  city (string): City,
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string): Postal code
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	

HTTP Status Code	Reason	Response Model
401	client unauthorized	

POST /external/{agencyid}/patrons/authenticate/v6 Authenticates a patron and returns the patron details.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
authenticationRequest	**credentials for patron to be authenticated**	body	Model Model Schema **AuthenticationRequest {** **pincode** (string): The pincode that belongs to the libraryCardNumber in plain text, **libraryCardNumber** (string): Identifies a libraryCard. This can be either a physical card or a CPR number that is used as a libraryCard **}**

Response Class

Model Model Schema

AuthenticatedPatronV6 {

patron (PatronV5, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,

authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV5 {

birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library

}

AddressV2 {

country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code

}

Period {

from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set

```
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/authenticate/v7

Authenticates a patron and returns the patron details.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
authenticationRequest	**credentials for patron to be authenticated**	body	Model Model Schema

Parameter	Description	Parameter Type	Data Type

			AuthenticationRequest {
			pincode (string): The pincode that belongs to the libraryCardNumber in plain text,
			libraryCardNumber (string): Identifies a libraryCard. This can be either a physical card or a CPR number that is used as a libraryCard
			}

Response Class

Model | Model Schema

AuthenticatedPatronV7 {

patron (PatronV6, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV6 {

birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
emailAddress (string, *optional*),
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,

defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
BlockStatus {
blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/authenticate/v8 Authenticates a patron and returns the patron details.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron.

If a patron is blocked the reason is available as a code:

- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
authenticationRequest	**credentials for patron to be authenticated**	body	Model Model Schema **AuthenticationRequest {** **pincode** (string): The pincode that belongs to the libraryCardNumber in plain text, **libraryCardNumber** (string): Identifies a libraryCard. This can be either a physical card or a CPR number that is used as a libraryCard **}**

Response Class

Model Model Schema

AuthenticatedPatronV8 {

patron (PatronV7, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV7 {

birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),

```
    onHold (Period, optional): If not set then the patron is not on hold,
    patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
    guardianVisibility (boolean),
    receiveEmail (boolean),
    blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
    receiveSms (boolean),
    tags (array[string], optional): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
    emailAddress (string, optional),
    notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
    phoneNumber (string, optional),
    name (string, optional),
    receivePostalMail (boolean),
    allowBookings (boolean, optional): True if the user is allowed to create bookings.,
    defaultInterestPeriod (integer): Length of default interest period in days,
    interests (array[InterestV1], optional),
    resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
InterestV1 {
    displayName (string): Display name of the interest,
    name (string): Name of the interest
```

}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/authenticate/v9 Authenticates a patron and returns a patron id, patron type and authentication status.

Implementation Notes

The returned patronId must be used by all subsequent service calls made on behalf of that patron. The full patron can be fetched with the operation corresponding to the patron type using the patron id and patron type.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
authenticationRequest	**credentials for patron to be authenticated**	body	Model Model Schema **AuthenticationRequest {** **pincode** (string): The pincode that belongs to the libraryCardNumber in plain text, **libraryCardNumber** (string): Identifies a libraryCard. This can be either a physical card or a CPR number that is used as a libraryCard **}**

Response Class

Model Model Schema

PatronAuthenticationResponse {

authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

,
patronId (integer),
patronType (string) = ['PERSON' or 'COMPANY' or 'LIBRARY' or 'GROUP']
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/company/{patronid}

Returns the patron details for company patron

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website | The codes are
- 'E': maximum amount of allowed debt exceeded | - 'D': deceased informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

CompanyPatron {

secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
contactPerson (string),
receiveSms (boolean),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
emailAddress (string, *optional*),
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
interests (array[InterestV1], *optional*),
cvr (string)

}

AddressV2 {

country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code

}

Period {

from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set

}

BlockStatus {

blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}
InterestV1 {
displayName (string): Display name of the interest,
name (string): Name of the interest
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/company/{patronid}/v2

Returns the patron details for company patron

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website | - 'E': maximum amount of allowed debt exceeded | - 'D': deceased The codes are informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

CompanyPatronV2 {

secondaryAddress (AddressV3, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV3, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
contactPerson (string),
phoneNumbers (array[PhoneNumberV1], *optional*),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
emailAddresses (array[EmailAddressV1], *optional*),
companyIdentifier (string),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
interests (array[InterestV1], *optional*)

}

AddressV3 {

country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code,
district (string): Dsitrict,
subDistrict (string): Subdistrict

}

Period {

from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set

```
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
PhoneNumberV1 {
  receiveNotification (boolean),
  phoneNumber (string)
}
EmailAddressV1 {
  emailAddress (string),
  receiveNotification (boolean)
}
InterestV1 {
  displayName (string): Display name of the interest,
  name (string): Name of the interest
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/company/{patronid}/v3

Returns the patron details for company patron

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website | The codes are
- 'E': maximum amount of allowed debt exceeded | - 'D': deceased informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model Model Schema

CompanyPatronV3 {

secondaryAddress (AddressV3, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV3, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
contactPerson (string),
phoneNumbers (array[PhoneNumberV1], *optional*),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
emailAddresses (array[EmailAddressV2], *optional*),
companyIdentifier (string),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
interests (array[InterestV1], *optional*)

}

AddressV3 {

country (string): Country,

```
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code,
    district (string): Dsitrict,
    subDistrict (string): Subdistrict
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
PhoneNumberV1 {
    receiveNotification (boolean),
    phoneNumber (string)
}
EmailAddressV2 {
    emailAddress (string),
    receiveNotification (boolean),
    verified (boolean)
}
InterestV1 {
    displayName (string): Display name of the interest,
    name (string): Name of the interest
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/company/{patronid}/v4

Returns the patron details for company patron

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website | The codes are
- 'E': maximum amount of allowed debt exceeded | - 'D': deceased informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

CompanyPatronV4 {

secondaryAddress (AddressV3, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV3, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
contactPerson (string),
phoneNumbers (array[PhoneNumberV1], *optional*),

```
  tags (array[string], optional): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
  notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
  emailAddresses (array[EmailAddressV2], optional),
  companyIdentifier (string),
  name (string, optional),
  receivePostalMail (boolean),
  allowBookings (boolean, optional): True if the user is allowed to create bookings.,
  defaultInterestPeriod (integer): Length of default interest period in days,
  patronNote (string, optional): Note to patron,
  interests (array[InterestV1], optional)
}
AddressV3 {
  country (string): Country,
  city (string): City,
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string): Postal code,
  district (string): Dsitrict,
  subDistrict (string): Subdistrict
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
PhoneNumberV1 {
  receiveNotification (boolean),
  phoneNumber (string)
}
EmailAddressV2 {
  emailAddress (string),
```

receiveNotification (boolean),
verified (boolean)
}
InterestV1 {
displayName (string): Display name of the interest,
name (string): Name of the interest
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/consent/{patronid}/v1 Get the list of given consents for a patron.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**owner of the consents**	path	integer

Response Class

Model Model Schema

ConsentV1 {
consentType (string): If you encounter another type of consent please add it to this documentation.

ConsentTypes:
- 'KEEP_HISTORICAL_LOAN_DATA'
- 'SYNC_WITH_NATIONAL_REGISTRY'

,

consentDate (string): Date that consent was given.
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
200	ok	
401	client unauthorized	
403	forbidden access to wrong agency	
404	no patron found for patronId	

GET /external/{agencyid}/patrons/group/{patronid}

Returns the patron details for a group patron

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website
- 'E': maximum amount of allowed debt exceeded
- 'D': deceased

The codes are informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

GroupPatron {

```
    secondaryAddress (AddressV2, optional),
    preferredLanguage (string, optional): Language in which the patron prefers the communication with the library to take place,
    preferredPickupBranch (string): ISIL of preferred pickup branch,
    address (AddressV2, optional),
    onHold (Period, optional): If not set then the patron is not on hold,
    patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
    receiveEmail (boolean),
    blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
    contactPerson (string),
    receiveSms (boolean),
    tags (array[string], optional): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
    emailAddress (string, optional),
    notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
    phoneNumber (string, optional),
    name (string, optional),
    receivePostalMail (boolean),
    allowBookings (boolean, optional): True if the user is allowed to create bookings.,
    defaultInterestPeriod (integer): Length of default interest period in days,
    interests (array[InterestV1], optional)
}
AddressV2 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
```

}
InterestV1 {
  displayName (string): Display name of the interest,
  name (string): Name of the interest
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/group/{patronid}/v2

Returns the patron details for a group patron

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website | The codes are
- 'E': maximum amount of allowed debt exceeded | - 'D': deceased informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

GroupPatronV2 {

```
    secondaryAddress (AddressV3, optional),
    preferredLanguage (string, optional): Language in which the patron prefers the communication with the library to take place,
    preferredPickupBranch (string): ISIL of preferred pickup branch,
    address (AddressV3, optional),
    onHold (Period, optional): If not set then the patron is not on hold,
    patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
    blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
    contactPerson (string),
    phoneNumbers (array[PhoneNumberV1], optional),
    tags (array[string], optional): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
    notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
    emailAddresses (array[EmailAddressV1], optional),
    name (string, optional),
    receivePostalMail (boolean),
    allowBookings (boolean, optional): True if the user is allowed to create bookings.,
    defaultInterestPeriod (integer): Length of default interest period in days,
    interests (array[InterestV1], optional)
}
AddressV3 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code,
    district (string): Dsitrict,
    subDistrict (string): Subdistrict
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
```

```
}
PhoneNumberV1 {
  receiveNotification (boolean),
  phoneNumber (string)
}
EmailAddressV1 {
  emailAddress (string),
  receiveNotification (boolean)
}
InterestV1 {
  displayName (string): Display name of the interest,
  name (string): Name of the interest
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/group/{patronid}/v3

Returns the patron details for a group patron

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website | The codes are
- 'E': maximum amount of allowed debt exceeded | - 'D': deceased informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**ISIL of the agency (e.g. DK-761500)**	path	string
`patronid`	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

GroupPatronV3 {
secondaryAddress (AddressV3, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV3, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
contactPerson (string),
phoneNumbers (array[PhoneNumberV1], *optional*),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
emailAddresses (array[EmailAddressV2], *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
interests (array[InterestV1], *optional*)
}
AddressV3 {
country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code,
district (string): Dsitrict,
subDistrict (string): Subdistrict

```
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
PhoneNumberV1 {
  receiveNotification (boolean),
  phoneNumber (string)
}
EmailAddressV2 {
  emailAddress (string),
  receiveNotification (boolean),
  verified (boolean)
}
InterestV1 {
  displayName (string): Display name of the interest,
  name (string): Name of the interest
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/group/{patronid}/v4

Returns the patron details for a group patron

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website | The codes are
- 'E': maximum amount of allowed debt exceeded | - 'D': deceased informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

GroupPatronV4 {

secondaryAddress (AddressV3, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV3, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
contactPerson (string),
phoneNumbers (array[PhoneNumberV1], *optional*),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
emailAddresses (array[EmailAddressV2], *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
patronNote (string, *optional*): Note to patron,

```
    interests (array[InterestV1], optional)
}
AddressV3 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code,
    district (string): Dsitrict,
    subDistrict (string): Subdistrict
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
PhoneNumberV1 {
    receiveNotification (boolean),
    phoneNumber (string)
}
EmailAddressV2 {
    emailAddress (string),
    receiveNotification (boolean),
    verified (boolean)
}
InterestV1 {
    displayName (string): Display name of the interest,
    name (string): Name of the interest
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/guardian/v1 Returns the patron details of all patrons for a guardian that have the guardian visibility enabled.

Implementation Notes

If the corresponding agency configuration through /external/v1/{agencyid}/configuration/loaner.guardianVisibility.enabled is not enabled, then response message 403 will be sent back.

If there are no patrons found, that have the guardian visibility enabled, for the given guardian CPR, then response message 404 will be sent back.

The returned patrons details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron. Note: This method can only be used for patrons who are people, and not e.g. Companies or Libraries.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
guardianPersonId		body	string

Response Class

Model Model Schema

AuthenticatedPatronV8 {

patron (PatronV7, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV7 {
birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
guardianVisibility (boolean),
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
emailAddress (string, *optional*),
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
interests (array[InterestV1], *optional*),
resident (boolean): True if the user is resident in the same municipality as the library

}

AddressV2 {
country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code

}
Period {
  from (string, *optional*): Open-ended if not set,
  to (string, *optional*): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
InterestV1 {
  displayName (string): Display name of the interest,
  name (string): Name of the interest
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
403	the guardians do not have access to patron details on this agency	
404	no patrons found that have the guardian as a financial resposible or that have the guardian visibility enabled	

POST /external/{agencyid}/patrons/guardian/v2 Returns the patron details of all patrons for a guardian that have the guardian visibility enabled.

Implementation Notes

If the corresponding agency configuration through /external/v1/{agencyid}/configuration/loaner.guardianVisibility.enabled is not enabled, then response message 403 will be sent back.

If there are no patrons found, that have the guardian visibility enabled, for the given guardian person identifier (e.g. CPR-number), then response message 404 will be sent back.

The returned patrons details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron. Note: This method can only be used for patrons who are people, and not e.g. Companies or Libraries.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
guardianPersonId		body	string

Response Class

Model | Model Schema

AuthenticatedPatronV10 {

patron (PatronV9, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV9 {

birthday (string, *optional*),
secondaryAddress (AddressV3, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV3, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,

```
    patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
    guardianVisibility (boolean),
    receiveEmail (boolean),
    blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
    keepLoanHistoricalData (boolean, optional): Patron consent to keep historical loans,
    receiveSms (boolean),
    tags (array[string], optional): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
    hasNationalRegistrySynchronizationConsent (boolean),
    emailAddress (string, optional),
    notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
    phoneNumber (string, optional),
    name (string, optional),
    receivePostalMail (boolean),
    allowBookings (boolean, optional): True if the user is allowed to create bookings.,
    defaultInterestPeriod (integer): Length of default interest period in days,
    interests (array[string], optional),
    resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV3 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code,
    district (string): Dsitrict,
    subDistrict (string): Subdistrict
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
403	the guardians do not have access to patron details on this agency	
404	no patrons found that have the guardian as a financial resposible or that have the guardian visibility enabled	

GET /external/{agencyid}/patrons/interests/v1 Returns a list of interests that are configured for the agency.

Implementation Notes

Interests can be added to patrons.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**agencyId ISIL of the agency (e.g. DK-761500)**	path	string

Response Class

Model | Model Schema

PatronInterestsV1 {
interests (Map)
}

Response Content Type application/json

GET /external/{agencyid}/patrons/languages/v1 Get the list of languages supported as preferred language of a patron.

Implementation Notes

Returns an array of string representation of the language codes.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string

Response Class

Model | Model Schema

array[string]

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/library/{patronid}

Returns the patron details for a library patron

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website | The codes are
- 'E': maximum amount of allowed debt exceeded | - 'D': deceased informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

LibraryPatron {
- **secondaryAddress** (AddressV2, *optional*),
- **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place,
- **preferredPickupBranch** (string): ISIL of preferred pickup branch,
- **address** (AddressV2, *optional*),
- **onHold** (Period, *optional*): If not set then the patron is not on hold,
- **patronId** (integer): Patron identifier to be used in subsequent service calls involving the patron,
- **receiveEmail** (boolean),
- **blockStatus** (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
- **receiveSms** (boolean),
- **tags** (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
- **emailAddress** (string, *optional*),
- **notificationProtocols** (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
- **phoneNumber** (string, *optional*),
- **name** (string, *optional*),
- **receivePostalMail** (boolean),
- **allowBookings** (boolean, *optional*): True if the user is allowed to create bookings.,
- **defaultInterestPeriod** (integer): Length of default interest period in days,
- **interests** (array[InterestV1], *optional*),
- **isil** (string): ISIL of the library

}

AddressV2 {
- **country** (string): Country,
- **city** (string): City,
- **street** (string): Street and number,
- **coName** (string): c/o name,

postalCode (string): Postal code
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
BlockStatus {
blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}
InterestV1 {
displayName (string): Display name of the interest,
name (string): Name of the interest
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/library/{patronid}/v2

Returns the patron details for a library patron

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website | The codes are
- 'E': maximum amount of allowed debt exceeded | - 'D': deceased informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

LibraryPatronV2 {

secondaryAddress (AddressV3, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV3, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
phoneNumbers (array[PhoneNumberV1], *optional*),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
emailAddresses (array[EmailAddressV1], *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
interests (array[InterestV1], *optional*),
isil (string)

}

AddressV3 {

country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code,
district (string): Dsitrict,

subDistrict (string): Subdistrict
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
BlockStatus {
blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}
PhoneNumberV1 {
receiveNotification (boolean),
phoneNumber (string)
}
EmailAddressV1 {
emailAddress (string),
receiveNotification (boolean)
}
InterestV1 {
displayName (string): Display name of the interest,
name (string): Name of the interest
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/library/{patronid}/v3

Returns the patron details for a library patron

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website | The codes are
- 'E': maximum amount of allowed debt exceeded | - 'D': deceased informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

LibraryPatronV3 {
secondaryAddress (AddressV3, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV3, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
phoneNumbers (array[PhoneNumberV1], *optional*),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
emailAddresses (array[EmailAddressV2], *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
interests (array[InterestV1], *optional*),
isil (string): ISIL of the library

```
}
AddressV3 {
  country (string): Country,
  city (string): City,
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string): Postal code,
  district (string): Dsitrict,
  subDistrict (string): Subdistrict
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
PhoneNumberV1 {
  receiveNotification (boolean),
  phoneNumber (string)
}
EmailAddressV2 {
  emailAddress (string),
  receiveNotification (boolean),
  verified (boolean)
}
InterestV1 {
  displayName (string): Display name of the interest,
  name (string): Name of the interest
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/library/{patronid}/v4

Returns the patron details for a library patron

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website | The codes are
- 'E': maximum amount of allowed debt exceeded | - 'D': deceased informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

LibraryPatronV4 {
- **secondaryAddress** (AddressV3, *optional*),
- **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place,
- **preferredPickupBranch** (string): ISIL of preferred pickup branch,
- **address** (AddressV3, *optional*),
- **onHold** (Period, *optional*): If not set then the patron is not on hold,
- **patronId** (integer): Patron identifier to be used in subsequent service calls involving the patron,
- **blockStatus** (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,

```
    phoneNumbers (array[PhoneNumberV1], optional),
    tags (array[string], optional): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
    notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
    emailAddresses (array[EmailAddressV2], optional),
    name (string, optional),
    receivePostalMail (boolean),
    allowBookings (boolean, optional): True if the user is allowed to create bookings.,
    defaultInterestPeriod (integer): Length of default interest period in days,
    patronNote (string, optional): Note to patron,
    interests (array[InterestV1], optional),
    isil (string): ISIL of the library
}
AddressV3 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code,
    district (string): Dsitrict,
    subDistrict (string): Subdistrict
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
PhoneNumberV1 {
    receiveNotification (boolean),
    phoneNumber (string)
}
EmailAddressV2 {
```

emailAddress (string),
receiveNotification (boolean),
verified (boolean)
}
InterestV1 {
displayName (string): Display name of the interest,
name (string): Name of the interest
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/notificationprotocols/v1 Get the list of notification protocols that is possible to set for a patron.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string

Response Class

Model | Model Schema

NotificationProtocol {
displayName (string): Display name of the notification protocol that is suitable to be shown to the patron.,
name (string): Name of the notification protocol
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/person/{patronid}

Returns the patron details for a person patron

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website | The codes are
- 'E': maximum amount of allowed debt exceeded | - 'D': deceased informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

PersonPatron {
birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
guardianVisibility (boolean),
receiveEmail (boolean),

```
  blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
  receiveSms (boolean),
  tags (array[string], optional): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
  emailAddress (string, optional),
  notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
  phoneNumber (string, optional),
  name (string, optional),
  receivePostalMail (boolean),
  allowBookings (boolean, optional): True if the user is allowed to create bookings.,
  defaultInterestPeriod (integer): Length of default interest period in days,
  interests (array[InterestV1], optional),
  resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
  country (string): Country,
  city (string): City,
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string): Postal code
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
InterestV1 {
  displayName (string): Display name of the interest,
  name (string): Name of the interest
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/person/{patronid}/v2

Returns the patron details for a person patron

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website | - 'E': maximum amount of allowed debt exceeded | - 'D': deceased | The codes are informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

PersonPatronV2 {
- **birthday** (string, *optional*),
- **secondaryAddress** (AddressV3, *optional*),
- **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place,
- **preferredPickupBranch** (string): ISIL of preferred pickup branch,
- **address** (AddressV3, *optional*),
- **onHold** (Period, *optional*): If not set then the patron is not on hold,
- **patronId** (integer): Patron identifier to be used in subsequent service calls involving the patron,

```
  guardianVisibility (boolean),
  blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
  phoneNumbers (array[PhoneNumberV1], optional),
  tags (array[string], optional): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
  notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
  emailAddresses (array[EmailAddressV1], optional),
  name (string, optional),
  receivePostalMail (boolean),
  allowBookings (boolean, optional): True if the user is allowed to create bookings.,
  defaultInterestPeriod (integer): Length of default interest period in days,
  interests (array[InterestV1], optional),
  resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV3 {
  country (string): Country,
  city (string): City,
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string): Postal code,
  district (string): Dsitrict,
  subDistrict (string): Subdistrict
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
PhoneNumberV1 {
  receiveNotification (boolean),
  phoneNumber (string)
```

```
}
EmailAddressV1 {
  emailAddress (string),
  receiveNotification (boolean)
}
InterestV1 {
  displayName (string): Display name of the interest,
  name (string): Name of the interest
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/person/{patronid}/v3

Returns the patron details for a person patron

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website | The codes are
- 'E': maximum amount of allowed debt exceeded | - 'D': deceased informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

PersonPatronV3 {
birthday (string, *optional*),
secondaryAddress (AddressV3, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV3, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
guardianVisibility (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
phoneNumbers (array[PhoneNumberV1], *optional*),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
emailAddresses (array[EmailAddressV2], *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
interests (array[InterestV1], *optional*),
resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV3 {
country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code,
district (string): Dsitrict,
subDistrict (string): Subdistrict
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set

```
}
BlockStatus {
   blockedReason (string): Reason code for block,
   blockedSince (string),
   message (string): Message about block
}
PhoneNumberV1 {
   receiveNotification (boolean),
   phoneNumber (string)
}
EmailAddressV2 {
   emailAddress (string),
   receiveNotification (boolean),
   verified (boolean)
}
InterestV1 {
   displayName (string): Display name of the interest,
   name (string): Name of the interest
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/{agencyid}/patrons/person/{patronid}/v4

Returns the patron details for a person patron

If a patron is blocked the reason is available as a code:
- 'O': library card stolen | - 'U': exclusion | - 'F': extended exclusion | - 'S': blocked by self service automaton | - 'W': self created at website | The codes are
- 'E': maximum amount of allowed debt exceeded | - 'D': deceased informational, and can be used for looking up end user messages by the client system.

Implementation Notes

However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**patron id of the patron**	path	integer

Response Class

Model | Model Schema

PersonPatronV4 {
- **birthday** (string, *optional*),
- **secondaryAddress** (AddressV3, *optional*),
- **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place,
- **preferredPickupBranch** (string): ISIL of preferred pickup branch,
- **address** (AddressV3, *optional*),
- **onHold** (Period, *optional*): If not set then the patron is not on hold,
- **patronId** (integer): Patron identifier to be used in subsequent service calls involving the patron,
- **guardianVisibility** (boolean),
- **blockStatus** (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
- **phoneNumbers** (array[PhoneNumberV1], *optional*),
- **tags** (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
- **notificationProtocols** (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
- **emailAddresses** (array[EmailAddressV2], *optional*),
- **name** (string, *optional*),
- **receivePostalMail** (boolean),
- **allowBookings** (boolean, *optional*): True if the user is allowed to create bookings.,
- **defaultInterestPeriod** (integer): Length of default interest period in days,
- **patronNote** (string, *optional*): Note to patron,
- **interests** (array[InterestV1], *optional*),
- **resident** (boolean): True if the user is resident in the same municipality as the library

```
}
AddressV3 {
  country (string): Country,
  city (string): City,
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string): Postal code,
  district (string): Dsitrict,
  subDistrict (string): Subdistrict
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
PhoneNumberV1 {
  receiveNotification (boolean),
  phoneNumber (string)
}
EmailAddressV2 {
  emailAddress (string),
  receiveNotification (boolean),
  verified (boolean)
}
InterestV1 {
  displayName (string): Display name of the interest,
  name (string): Name of the interest
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/preauthenticated/patronId/v1

Returns the patron details of a patron that the client has pre-authenticated using a third party.

Implementation Notes

Patrons can be looked up based on their patronId.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronId	**patron id of the patron**	body	integer

Response Class

Model | Model Schema

AuthenticatedPatronV6 {

patron (PatronV5, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,

authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:

- 'VALID': successfully authenticated

```
    - 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
    - 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts
}
PatronV5 {
    birthday (string, optional),
    secondaryAddress (AddressV2, optional),
    preferredLanguage (string, optional): Language in which the patron prefers the communication with the library to take place,
    preferredPickupBranch (string): ISIL of preferred pickup branch,
    address (AddressV2, optional),
    onHold (Period, optional): If not set then the patron is not on hold,
    patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
    receiveEmail (boolean),
    blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
    receiveSms (boolean),
    emailAddress (string, optional),
    notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
    phoneNumber (string, optional),
    name (string, optional),
    receivePostalMail (boolean),
    allowBookings (boolean, optional): True if the user is allowed to create bookings.,
    defaultInterestPeriod (integer): Length of default interest period in days,
    resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/preauthenticated/patronId/v2

Returns the patron details of a patron that the client has pre-authenticated using a third party.

Implementation Notes

Patrons can be looked up based on their patronId.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronId	**patron id of the patron**	body	integer

Response Class

Model | Model Schema

AuthenticatedPatronV7 {

patron (PatronV6, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV6 {

birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
emailAddress (string, *optional*),
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library

}

AddressV2 {

country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code

```
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/preauthenticated/patronId/v3

Returns the patron details of a patron that the client has pre-authenticated using a third party.

Implementation Notes

Patrons can be looked up based on their patronId.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronId	**patron id of the patron**	body	integer

Response Class

Model | Model Schema

AuthenticatedPatronV8 {

patron (PatronV7, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV7 {

birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
guardianVisibility (boolean),
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
emailAddress (string, *optional*),
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,

defaultInterestPeriod (integer): Length of default interest period in days,
interests (array[InterestV1], *optional*),
resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
BlockStatus {
blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}
InterestV1 {
displayName (string): Display name of the interest,
name (string): Name of the interest
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/preauthenticated/patronId/v4

Returns a patron id, patron type and authentication status for a patron that the client has pre-authenticated using a third party.

Implementation Notes

Patrons can be looked up based on their patronId. The returned patronId must be used by all subsequent service calls made on behalf of that patron. The full patron can be fetched with the operation corresponding to the patron type using the patron id and patron type.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronId	**patron id of the patron**	body	integer

Response Class

Model | Model Schema

PatronAuthenticationResponse {

authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

,
patronId (integer),
patronType (string) = ['PERSON' or 'COMPANY' or 'LIBRARY' or 'GROUP']

}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/preauthenticated/unic/v3

Returns the patron details of a patron that the client has pre-authenticated using UNIC.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron. Note: This method can only be used for patrons who are persons, and not e.g. Companies or Libraries.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website
- 'E': maximum amount of allowed debt exceeded
- 'D': deceased

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**ISIL of the agency (e.g. DK-761500)**	path	string
`unicUsername`	**UNIC username of the patron**	body	string

Response Class

Model | Model Schema

AuthenticatedPatronV3 {

authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,

patron (PatronV3, *optional*): Only available if patron exists in FBS and was succesfully authenticated.

}

PatronV3 {

birthday (string, *optional*),

secondaryAddress (Address, *optional*),

preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,

preferredPickupBranch (string): ISIL of preferred pickup branch,

address (Address, *optional*),

onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library
}
Address {
country (string),
city (string),
street (string): Street and number,
coName (string): c/o name,
postalCode (string)
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
BlockStatus {
blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	

HTTP Status Code	Reason	Response Model
401	client unauthorized	

POST /external/{agencyid}/patrons/preauthenticated/unic/v4

Returns the patron details of a patron that the client has pre-authenticated using UNIC.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron. Note: This method can only be used for patrons who are persons, and not e.g. Companies or Libraries.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
unicUsername	**UNIC username of the patron**	body	string

Response Class

Model | Model Schema

AuthenticatedPatronV4 {

authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,

patron (PatronV4, *optional*): Only available if patron exists in FBS and was succesfully authenticated.

}

PatronV4 {

```
    birthday (string, optional),
    secondaryAddress (AddressV2, optional),
    preferredLanguage (string, optional): Language in which the patron prefers the communication with the library to take place,
    preferredPickupBranch (string): ISIL of preferred pickup branch,
    address (AddressV2, optional),
    onHold (Period, optional): If not set then the patron is not on hold,
    patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
    receiveEmail (boolean),
    blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
    receiveSms (boolean),
    emailAddress (string, optional),
    phoneNumber (string, optional),
    name (string, optional),
    receivePostalMail (boolean),
    allowBookings (boolean, optional): True if the user is allowed to create bookings.,
    defaultInterestPeriod (integer): Length of default interest period in days,
    resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/preauthenticated/unic/v5

Returns the patron details of a patron that the client has pre-authenticated using UNIC.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron. Note: This method can only be used for patrons who are persons, and not e.g. Companies or Libraries.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**ISIL of the agency (e.g. DK-761500)**	path	string
`unicUsername`	**UNIC username of the patron**	body	string

Response Class

Model Model Schema

AuthenticatedPatronV5 {

patron (PatronV4, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,

```
  authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
  - 'VALID': successfully authenticated
  - 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
  - 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts
}
PatronV4 {
  birthday (string, optional),
  secondaryAddress (AddressV2, optional),
  preferredLanguage (string, optional): Language in which the patron prefers the communication with the library to take place,
  preferredPickupBranch (string): ISIL of preferred pickup branch,
  address (AddressV2, optional),
  onHold (Period, optional): If not set then the patron is not on hold,
  patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
  receiveEmail (boolean),
  blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
  receiveSms (boolean),
  emailAddress (string, optional),
  phoneNumber (string, optional),
  name (string, optional),
  receivePostalMail (boolean),
  allowBookings (boolean, optional): True if the user is allowed to create bookings.,
  defaultInterestPeriod (integer): Length of default interest period in days,
  resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
  country (string): Country,
  city (string): City,
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string): Postal code
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
```

blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/preauthenticated/unic/v6

Returns the patron details of a patron that the client has pre-authenticated using UNIC (DEPRECATED).

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron. Note: This method can only be used for patrons who are persons, and not e.g. Companies or Libraries.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
unicUsername	**UNIC username of the patron**	body	string

Response Class

Model | Model Schema

AuthenticatedPatronV6 {

patron (PatronV5, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV5 {

birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library

}

AddressV2 {

country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code

```
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/preauthenticated/v10

Returns a patron id, patron type and authentication status for a patron that the client has pre-authenticated using a third party.

Implementation Notes

Patrons can be looked up based on a patronIdentifier number (e.g. CPR, Library card, UNI-Login). The returned patronId must be used by all subsequent service calls made on behalf of that patron. The full patron can be fetched with the operation corresponding to the patron type using the patron id and patron type.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronIdentifier	**Given patron identifier**	body	string

Response Class

Model | Model Schema

PatronAuthenticationResponse {
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

,
patronId (integer),
patronType (string) = ['PERSON' or 'COMPANY' or 'LIBRARY' or 'GROUP']
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/preauthenticated/v3

Returns the patron details of a patron that the client has pre-authenticated using a third party.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron. Note: Whis method can only be used for patrons who are people, and not e.g. Companies or Libraries.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**ISIL of the agency (e.g. DK-761500)**	path	string
`cprNumber`	**CPR-number of the patron**	body	string

Response Class

Model | Model Schema

AuthenticatedPatronV3 {

authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,
patron (PatronV3, *optional*): Only available if patron exists in FBS and was succesfully authenticated.
}

PatronV3 {

birthday (string, *optional*),
secondaryAddress (Address, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (Address, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library

```
}
Address {
  country (string),
  city (string),
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string)
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/preauthenticated/v4

Returns the patron details of a patron that the client has pre-authenticated using a third party.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron. Note: Whis method can only be used for patrons who are people, and not e.g. Companies or Libraries.

If a patron is blocked the reason is available as a code:

- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
cprNumber	**CPR-number of the patron**	body	string

Response Class

Model | Model Schema

AuthenticatedPatronV4 {

authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,

patron (PatronV4, *optional*): Only available if patron exists in FBS and was succesfully authenticated.

}

PatronV4 {

birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),

```
    allowBookings (boolean, optional): True if the user is allowed to create bookings.,
    defaultInterestPeriod (integer): Length of default interest period in days,
    resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/preauthenticated/v5

Returns the patron details of a patron that the client has pre-authenticated using a third party.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron. Note: Whis method can only be used for patrons who are people, and not e.g. Companies or Libraries.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
cprNumber	**CPR-number of the patron**	body	string

Response Class

Model | Model Schema

AuthenticatedPatronV5 {

patron (PatronV4, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV4 {

birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),

blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
BlockStatus {
blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/preauthenticated/v6

Returns the patron details of a patron that the client has pre-authenticated using a third party (DEPRECATED).

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron. Note: Whis method can only be used for patrons who are people, and not e.g. Companies or Libraries.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
cprNumber	**CPR-number of the patron**	body	string

Response Class

Model | Model Schema

AuthenticatedPatronV6 {

patron (PatronV5, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV5 {

birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,

preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
BlockStatus {
blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/preauthenticated/v7

Returns the patron details of a patron that the client has pre-authenticated using a third party.

Implementation Notes

Patrons can be looked up based on a patronIdentifier number (e.g. CPR, Library card, UNI-Login). The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron. Note: This method can only be used for patrons who are people, and not e.g. Companies or Libraries.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronIdentifier	**Given patron identifier**	body	string

Response Class

Model | Model Schema

AuthenticatedPatronV6 {

patron (PatronV5, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated

- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV5 {

birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library

}

AddressV2 {

country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code

}

Period {

from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set

}

BlockStatus {

blockedReason (string): Reason code for block,

```
  blockedSince (string),
  message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/preauthenticated/v8

Returns the patron details of a patron that the client has pre-authenticated using a third party.

Implementation Notes

Patrons can be looked up based on a patronIdentifier number (e.g. CPR, Library card, UNI-Login). The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron. Note: This method can only be used for patrons who are people, and not e.g. Companies or Libraries.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronIdentifier	**Given patron identifier**	body	string

Response Class

Model | Model Schema

AuthenticatedPatronV7 {
patron (PatronV6, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV6 {
birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
emailAddress (string, *optional*),
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library

}

AddressV2 {
country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,

postalCode (string): Postal code
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
BlockStatus {
blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/preauthenticated/v9

Returns the patron details of a patron that the client has pre-authenticated using a third party.

Implementation Notes

Patrons can be looked up based on a patronIdentifier number (e.g. CPR, Library card, UNI-Login). The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron. Note: Whis method can only be used for patrons who are people, and not e.g. Companies or Libraries.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronIdentifier	**Given patron identifier**	body	string

Response Class

Model | Model Schema

AuthenticatedPatronV8 {

patron (PatronV7, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV7 {

birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
guardianVisibility (boolean),
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
emailAddress (string, *optional*),
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*),

```
    name (string, optional),
    receivePostalMail (boolean),
    allowBookings (boolean, optional): True if the user is allowed to create bookings.,
    defaultInterestPeriod (integer): Length of default interest period in days,
    interests (array[InterestV1], optional),
    resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
InterestV1 {
    displayName (string): Display name of the interest,
    name (string): Name of the interest
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/{agencyid}/patrons/resetPincode/v1 Send a reset pincode email to the patron with the provided emailaddress.

Implementation Notes

The response is successful, if an email was successfully sent to the patron to reset their pincode.

If the supplied emailaddress has an invalid format, then response message 400 will be sent back.

If multiple patrons within the supplied agency have the same emailaddress, then response message 400 will be sent back.

If the emailadress does not correspond to any patron, then response message 404 will be sent back.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
emailAddress	**emailAddress belonging to the patron**	body	string

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request - invalid format or multiple patrons found for emailaddress	
401	client unauthorized	
403	forbidden access to wrong agency	
404	no patron found for emailAddress	

POST /external/{agencyid}/patrons/updatePincode/v1 Using the UUID provided through a reset pincode email, update the pincode of the patron.

Implementation Notes

The response is successful, if the pincode was successfully updated.

If the supplied pincode is invalid, blank or too long, then response message 400 will be sent back.

If the supplied UUID is invalid, blank or not a valid format, then response message 400 will be sent back.

If the UUID has expired or cannot be found, then response message 404 will be sent back.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
updatePincode	**Request to update patron pincode**	body	Model | Model Schema **UpdatePincodeRequestV1 {** **pincode** (string): The new pincode for the patron, **uuid** (string): UUID provided to patron in email when resetting pincode **}**

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request - invalid format for pincode or UUID	
401	client unauthorized	
403	forbidden access to wrong agency	
404	UUID not found	

POST /external/{agencyid}/patrons/v10 Create a new patron who is a person.

Implementation Notes

When a patron doesn't have a patron account in the library system, but logs in using a trusted authentication source (e.g NemId), the patron account can be created using this service. Name and address will be automatically fetched from configured person registry, and cannot be supplied by the client. If the configured person registry is not authorized to provide information about the patron, then response message 404 will be sent back

Multiple email addresses and phone numbers can be stored for a patron.

If multiple email addresses are supplied having receiveNotification as true, then only one of them will be randomly stored as preferred and the rest will be stored as not preferred.

If multiple phone numbers are supplied having receiveNotification as true, then only one of them will be randomly stored as preferred and the rest will be stored as not preferred.

This version includes a possibility of including a BlockStatus in the request. The reason of the blockstatus must be one of the following one-letter choices

- 'O': library card stolen
- 'F': extended exclusion
- 'U': exclusion
- 'S': blocked by self service automaton
- 'W': self created at website
- 'E': maximum amount of allowed debt exceeded
- 'D': deceased

If the request includes a blockstatus for the patron, and the blockedSince date is not given or in the wrong format, or the blockedReason is not one of the one-letter choices shown above, then response message 400 will be sent back.

Setting nationalRegistryConsent to true will allow the data of the created patron to be synchronized with the Norwegian national registry.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
createPatronRequest	**the patron to be created**	body	Model Model Schema **CreatePatronRequestV8 {** **pincode** (string, *optional*), **patron** (PatronSettingsV6), **nationalRegistryConsent** (boolean, *optional*), **personIdentifier** (string), **blockStatusRequest** (BlockStatusRequest, *optional*) **}** **PatronSettingsV6 {** **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place If left empty default library language will be used,

Parameter	Description	Parameter Type	Data Type
			notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included., **emailAddresses** (array[EmailAddressV1], *optional*): Existing email addresses are overwritten with these values If left empty existing email addresses are deleted, **preferredPickupBranch** (string): ISIL-number of preferred pickup branch, **onHold** (Period, *optional*): If not set then the patron is not on hold, **guardianVisibility** (boolean), **receivePostalMail** (boolean), **interests** (array[string], *optional*): A list of interests of the patron., **phoneNumbers** (array[PhoneNumberV1], *optional*): Existing phonenumbers are overwritten with these values If left empty existing phonenumbers are deleted **}** **EmailAddressV1 {** **emailAddress** (string), **receiveNotification** (boolean) **}** **Period {** **from** (string, *optional*): Open-ended if not set, **to** (string, *optional*): Open-ended if not set **}** **PhoneNumberV1 {** **receiveNotification** (boolean), **phoneNumber** (string) **}** **BlockStatusRequest {** **blockedReason** (string): Reason code for block, **blockedSince** (string): The date from which the patron should be blocked. The dateformat is YYYY-MM-DD, so 9th of March 2023 is written 2023-03-09. **}**

Parameter	Description	Parameter Type	Data Type

Response Class

Model | Model Schema

AuthenticatedPatronV10 {
patron (PatronV9, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}
PatronV9 {
birthday (string, *optional*),
secondaryAddress (AddressV3, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV3, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
guardianVisibility (boolean),
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
keepLoanHistoricalData (boolean, *optional*): Patron consent to keep historical loans,
receiveSms (boolean),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
hasNationalRegistrySynchronizationConsent (boolean),
emailAddress (string, *optional*),
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
interests (array[string], *optional*),
resident (boolean): True if the user is resident in the same municipality as the library

```
}
AddressV3 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code,
    district (string): Dsitrict,
    subDistrict (string): Subdistrict
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	Data not found	

POST /external/{agencyid}/patrons/v3

Create a new patron who is a person.

Implementation Notes

When a patron doesn't have a patron account in the library system, but logs in using a trusted authentication source (e.g NemId), the patron account can be created using this service. Name and address will be automatically fetched from CPR-Registry, and cannot be supplied by the client. If the CPR-Registry is not authorized to provide information about the patron, then repsonse message 404 will be sent back

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
createPatronRequest	**the patron to be created**	body	Model | Model Schema **CreatePatronRequestV3 {** **cprNumber** (string), **pincode** (string), **patron** (PatronSettingsV3) **}** **PatronSettingsV3 {** **emailAddress** (string, *optional*): Required if patron should receive email notifications Existing email addresses are overwritten with this value If left empty existing email addresses are deleted, **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place If left empty default library language will be used, **phoneNumber** (string, *optional*): Required if patron should receive SMS notifications Existing phonenumbers are overwritten with this value If left empty existing phonenumbers are deleted, **preferredPickupBranch** (string): ISIL-number of preferred pickup branch,

Parameter	Description	Parameter Type	Data Type
			onHold (Period, *optional*): If not set then the patron is not on hold, **receiveEmail** (boolean), **receivePostalMail** (boolean), **receiveSms** (boolean) **}** **Period {** **from** (string, *optional*): Open-ended if not set, **to** (string, *optional*): Open-ended if not set **}**

Response Class

Model | Model Schema

AuthenticatedPatronV3 {
authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,
patron (PatronV3, *optional*): Only available if patron exists in FBS and was succesfully authenticated.
}
PatronV3 {
birthday (string, *optional*),
secondaryAddress (Address, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (Address, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,

resident (boolean): True if the user is resident in the same municipality as the library
}
Address {
country (string),
city (string),
street (string): Street and number,
coName (string): c/o name,
postalCode (string)
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
BlockStatus {
blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	Data not found	

POST /external/{agencyid}/patrons/v4 Create a new patron who is a person.

Implementation Notes

When a patron doesn't have a patron account in the library system, but logs in using a trusted authentication source (e.g NemId), the patron account can be created using this service. Name and address will be automatically fetched from CPR-Registry, and cannot be supplied by the client. If the CPR-Registry is not authorized to provide information about the patron, then repsonse message 404 will be sent back

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
createPatronRequest	**the patron to be created**	body	Model | Model Schema **CreatePatronRequestV3 {** **cprNumber** (string), **pincode** (string), **patron** (PatronSettingsV3) **}** **PatronSettingsV3 {** **emailAddress** (string, *optional*): Required if patron should receive email notifications Existing email addresses are overwritten with this value If left empty existing email addresses are deleted, **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place If left empty default library language will be used, **phoneNumber** (string, *optional*): Required if patron should receive SMS notifications Existing phonenumbers are overwritten with this value If left empty existing phonenumbers are deleted, **preferredPickupBranch** (string): ISIL-number of preferred pickup branch, **onHold** (Period, *optional*): If not set then the patron is not on hold,

Parameter	Description	Parameter Type	Data Type
			receiveEmail (boolean), **receivePostalMail** (boolean), **receiveSms** (boolean) **}** **Period {** **from** (string, *optional*): Open-ended if not set, **to** (string, *optional*): Open-ended if not set **}**

Response Class

Model | Model Schema

AuthenticatedPatronV4 {

authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,
patron (PatronV4, *optional*): Only available if patron exists in FBS and was succesfully authenticated.

}

PatronV4 {

birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library

}
AddressV2 {
  country (string): Country,
  city (string): City,
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string): Postal code
}
Period {
  from (string, *optional*): Open-ended if not set,
  to (string, *optional*): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	Data not found	

POST /external/{agencyid}/patrons/v5 Create a new patron who is a person.

Implementation Notes

When a patron doesn't have a patron account in the library system, but logs in using a trusted authentication source (e.g NemId), the patron account can be created using this service. Name and address will be automatically fetched from CPR-Registry, and cannot be supplied by the client. If the CPR-Registry is not authorized to provide information about the patron, then repsonse message 404 will be sent back

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
createPatronRequest	**the patron to be created**	body	Model | Model Schema **CreatePatronRequestV4 {** **cprNumber** (string), **pincode** (string), **patron** (PatronSettingsV4) **}** **PatronSettingsV4 {** **emailAddress** (string, *optional*): Required if patron should receive email notifications Existing email addresses are overwritten with this value If left empty existing email addresses are deleted, **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place If left empty default library language will be used, **notificationProtocols** (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included., **phoneNumber** (string, *optional*): Required if patron should receive SMS notifications Existing phonenumbers are overwritten with this value If left empty existing phonenumbers are deleted, **preferredPickupBranch** (string): ISIL-number of preferred pickup branch,

Parameter	Description	Parameter Type	Data Type
			onHold (Period, *optional*): If not set then the patron is not on hold, receiveEmail (boolean), receivePostalMail (boolean), receiveSms (boolean) } **Period {** from (string, *optional*): Open-ended if not set, to (string, *optional*): Open-ended if not set }

Response Class

Model | Model Schema

AuthenticatedPatronV6 {

patron (PatronV5, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV5 {

birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*),
name (string, *optional*),

```
    receivePostalMail (boolean),
    allowBookings (boolean, optional): True if the user is allowed to create bookings.,
    defaultInterestPeriod (integer): Length of default interest period in days,
    resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	Data not found	

POST /external/{agencyid}/patrons/v6 Create a new patron who is a person.

Implementation Notes

When a patron doesn't have a patron account in the library system, but logs in using a trusted authentication source (e.g NemId), the patron account can be created using this service. Name and address will be automatically fetched from CPR-Registry, and cannot be supplied by the client. If the CPR-Registry is not authorized to provide information about the patron, then repsonse message 404 will be sent back

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**ISIL of the agency (e.g. DK-761500)**	path	string
`createPatronRequest`	**the patron to be created**	body	Model | Model Schema **CreatePatronRequestV4 {** **cprNumber** (string), **pincode** (string), **patron** (PatronSettingsV4) **}** **PatronSettingsV4 {** **emailAddress** (string, *optional*): Required if patron should receive email notifications Existing email addresses are overwritten with this value If left empty existing email addresses are deleted, **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place If left empty default library language will be used, **notificationProtocols** (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,

Parameter	Description	Parameter Type	Data Type
			phoneNumber (string, *optional*): Required if patron should receive SMS notifications Existing phonenumbers are overwritten with this value If left empty existing phonenumbers are deleted, **preferredPickupBranch** (string): ISIL-number of preferred pickup branch, **onHold** (Period, *optional*): If not set then the patron is not on hold, **receiveEmail** (boolean), **receivePostalMail** (boolean), **receiveSms** (boolean) **}** **Period {** **from** (string, *optional*): Open-ended if not set, **to** (string, *optional*): Open-ended if not set **}**

Response Class

Model | Model Schema

AuthenticatedPatronV7 {

patron (PatronV6, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,

authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:

- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}

PatronV6 {

birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),

blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
tags (array[string], *optional*): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
emailAddress (string, *optional*),
notificationProtocols (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
country (string): Country,
city (string): City,
street (string): Street and number,
coName (string): c/o name,
postalCode (string): Postal code
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
BlockStatus {
blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	

HTTP Status Code	Reason	Response Model
401	client unauthorized	
404	Data not found	

POST /external/{agencyid}/patrons/v7 Create a new patron who is a person.

Implementation Notes

When a patron doesn't have a patron account in the library system, but logs in using a trusted authentication source (e.g NemId), the patron account can be created using this service. Name and address will be automatically fetched from CPR-Registry, and cannot be supplied by the client. If the CPR-Registry is not authorized to provide information about the patron, then repsonse message 404 will be sent back

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
createPatronRequest	**the patron to be created**	body	Model Model Schema **CreatePatronRequestV5 {** **cprNumber** (string), **pincode** (string), **patron** (PatronSettingsV5) **}** **PatronSettingsV5 {** **emailAddress** (string, *optional*): Required if patron should receive email notifications Existing email

Parameter	Description	Parameter Type	Data Type
			addresses are overwritten with this value If left empty existing email addresses are deleted, **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place If left empty default library language will be used, **notificationProtocols** (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included., **phoneNumber** (string, *optional*): Required if patron should receive SMS notifications Existing phonenumbers are overwritten with this value If left empty existing phonenumbers are deleted, **preferredPickupBranch** (string): ISIL-number of preferred pickup branch, **onHold** (Period, *optional*): If not set then the patron is not on hold, **guardianVisibility** (boolean), **receiveEmail** (boolean), **receivePostalMail** (boolean), **interests** (array[string], *optional*): A list of interests of the patron., **receiveSms** (boolean) **}** **Period {** **from** (string, *optional*): Open-ended if not set, **to** (string, *optional*): Open-ended if not set **}**

Response Class

Model | Model Schema

AuthenticatedPatronV8 {

patron (PatronV7, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,

authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:

- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

```
}
PatronV7 {
  birthday (string, optional),
  secondaryAddress (AddressV2, optional),
  preferredLanguage (string, optional): Language in which the patron prefers the communication with the library to take place,
  preferredPickupBranch (string): ISIL of preferred pickup branch,
  address (AddressV2, optional),
  onHold (Period, optional): If not set then the patron is not on hold,
  patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
  guardianVisibility (boolean),
  receiveEmail (boolean),
  blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
  receiveSms (boolean),
  tags (array[string], optional): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
  emailAddress (string, optional),
  notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
  phoneNumber (string, optional),
  name (string, optional),
  receivePostalMail (boolean),
  allowBookings (boolean, optional): True if the user is allowed to create bookings.,
  defaultInterestPeriod (integer): Length of default interest period in days,
  interests (array[InterestV1], optional),
  resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
  country (string): Country,
  city (string): City,
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string): Postal code
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
InterestV1 {
  displayName (string): Display name of the interest,
  name (string): Name of the interest
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	Data not found	

POST /external/{agencyid}/patrons/v8

Create a new patron who is a person.

Implementation Notes

When a patron doesn't have a patron account in the library system, but logs in using a trusted authentication source (e.g NemId), the patron account can be created using this service. Name and address will be automatically fetched from CPR-Registry, and cannot be supplied by the client. If the CPR-Registry is not authorized to provide information about the patron, then response message 404 will be sent back

This version includes a possibility of including a BlockStatus in the request. The reason of the blockstatus must be one of the following one-letter choices

- 'O': library card stolen
- 'F': extended exclusion
- 'U': exclusion
- 'S': blocked by self service automaton
- 'W': self created at website
- 'E': maximum amount of allowed debt exceeded
- 'D': deceased

If the request includes a blockstatus for the patron, and the blockedSince date is not given or in the wrong format, or the blockedReason is not one of the one-letter choices shown above, then response message 400 will be sent back.

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**ISIL of the agency (e.g. DK-761500)**	path	string
`createPatronRequest`	**the patron to be created**	body	Model Model Schema **CreatePatronRequestV6 {** **cprNumber** (string), **pincode** (string), **patron** (PatronSettingsV5), **blockStatusRequest** (BlockStatusRequest, *optional*) **}** **PatronSettingsV5 {** **emailAddress** (string, *optional*): Required if patron should receive email notifications Existing email addresses are overwritten with this value If left empty existing email addresses are deleted, **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place If left empty default library language will be used, **notificationProtocols** (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included., **phoneNumber** (string, *optional*): Required if patron should receive SMS notifications Existing phonenumbers are overwritten with this value If left empty existing phonenumbers are deleted, **preferredPickupBranch** (string): ISIL-number of preferred pickup branch, **onHold** (Period, *optional*): If not set then the patron is not on hold, **guardianVisibility** (boolean), **receiveEmail** (boolean), **receivePostalMail** (boolean),

Parameter	Description	Parameter Type	Data Type
			interests (array[string], *optional*): A list of interests of the patron., **receiveSms** (boolean) } **Period {** **from** (string, *optional*): Open-ended if not set, **to** (string, *optional*): Open-ended if not set } **BlockStatusRequest {** **blockedReason** (string): Reason code for block, **blockedSince** (string): The date from which the patron should be blocked. The dateformat is YYYY-MM-DD, so 9th of March 2023 is written 2023-03-09. }

Response Class

Model | Model Schema

AuthenticatedPatronV9 {
patron (PatronV8, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts
}
PatronV8 {
birthday (string, *optional*),
secondaryAddress (AddressV2, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV2, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
guardianVisibility (boolean),
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,

```
    keepLoanHistoricalData (boolean, optional): Patron consent to keep historical loans,
    receiveSms (boolean),
    tags (array[string], optional): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
    hasNationalRegistrySynchronizationConsent (boolean),
    emailAddress (string, optional),
    notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
    phoneNumber (string, optional),
    name (string, optional),
    receivePostalMail (boolean),
    allowBookings (boolean, optional): True if the user is allowed to create bookings.,
    defaultInterestPeriod (integer): Length of default interest period in days,
    interests (array[string], optional),
    resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV2 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	Data not found	

POST /external/{agencyid}/patrons/v9 Create a new patron who is a person.

Implementation Notes

When a patron doesn't have a patron account in the library system, but logs in using a trusted authentication source (e.g NemId), the patron account can be created using this service. Name and address will be automatically fetched from configured person registry, and cannot be supplied by the client. If the configured person registry is not authorized to provide information about the patron, then response message 404 will be sent back

Multiple email addresses and phone numbers can be stored for a patron.

If multiple email addresses are supplied having receiveNotification as true, then only one of them will be randomly stored as preferred and the rest will be stored as not preferred.

If multiple phone numbers are supplied having receiveNotification as true, then only one of them will be randomly stored as preferred and the rest will be stored as not preferred.

This version includes a possibility of including a BlockStatus in the request. The reason of the blockstatus must be one of the following one-letter choices

- 'O': library card stolen
- 'F': extended exclusion
- 'U': exclusion
- 'S': blocked by self service automaton
- 'W': self created at website
- 'E': maximum amount of allowed debt exceeded
- 'D': deceased

If the request includes a blockstatus for the patron, and the blockedSince date is not given or in the wrong format, or the blockedReason is not one of the one-letter choices shown above, then response message 400 will be sent back.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
createPatronRequest	**the patron to be created**	body	Model | Model Schema **CreatePatronRequestV7 {** **pincode** (string), **patron** (PatronSettingsV6), **personIdentifier** (string), **blockStatusRequest** (BlockStatusRequest, *optional*) **}** **PatronSettingsV6 {** **preferredLanguage** (string, *optional*): Language in which the patron prefers the communication with the library to take place If left empty default library language will be used, **notificationProtocols** (array[string], *optional*): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included., **emailAddresses** (array[EmailAddressV1], *optional*): Existing email addresses are overwritten with these values If left empty existing email addresses are deleted, **preferredPickupBranch** (string): ISIL-number of preferred pickup branch, **onHold** (Period, *optional*): If not set then the patron is not on hold, **guardianVisibility** (boolean), **receivePostalMail** (boolean), **interests** (array[string], *optional*): A list of interests of the patron., **phoneNumbers** (array[PhoneNumberV1], *optional*): Existing phonenumbers are overwritten with these values If left empty existing phonenumbers are deleted **}** **EmailAddressV1 {** **emailAddress** (string), **receiveNotification** (boolean)

Parameter	Description	Parameter Type	Data Type
			} **Period {** **from** (string, *optional*): Open-ended if not set, **to** (string, *optional*): Open-ended if not set } **PhoneNumberV1 {** **receiveNotification** (boolean), **phoneNumber** (string) } **BlockStatusRequest {** **blockedReason** (string): Reason code for block, **blockedSince** (string): The date from which the patron should be blocked. The dateformat is YYYY-MM-DD, so 9th of March 2023 is written 2023-03-09. }

Response Class

Model | Model Schema

AuthenticatedPatronV10 {
patron (PatronV9, *optional*): Only available if patron exists in FBS and was succesfully authenticated.,
authenticateStatus (string) = ['VALID' or 'INVALID' or 'LOANER_LOCKED_OUT']: AuthenticateStatus:
- 'VALID': successfully authenticated
- 'INVALID': either the user is not known in the system, or an invalid combination of authentication parameters has been used.
- 'LOANER_LOCKED_OUT': user has been blocked temporary because of too many failed login attempts

}
PatronV9 {
birthday (string, *optional*),
secondaryAddress (AddressV3, *optional*),
preferredLanguage (string, *optional*): Language in which the patron prefers the communication with the library to take place,
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (AddressV3, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
guardianVisibility (boolean),
receiveEmail (boolean),

```
  blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
  keepLoanHistoricalData (boolean, optional): Patron consent to keep historical loans,
  receiveSms (boolean),
  tags (array[string], optional): Tags associated with the patron. Can be used to e.g. give permissions to patron in external ressources.,
  hasNationalRegistrySynchronizationConsent (boolean),
  emailAddress (string, optional),
  notificationProtocols (array[string], optional): Notification protocols that the patron want to receive notification on. SMS and EMAIL are not included.,
  phoneNumber (string, optional),
  name (string, optional),
  receivePostalMail (boolean),
  allowBookings (boolean, optional): True if the user is allowed to create bookings.,
  defaultInterestPeriod (integer): Length of default interest period in days,
  interests (array[string], optional),
  resident (boolean): True if the user is resident in the same municipality as the library
}
AddressV3 {
  country (string): Country,
  city (string): City,
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string): Postal code,
  district (string): Dsitrict,
  subDistrict (string): Subdistrict
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	Data not found	

POST /external/{agencyid}/patrons/withGuardian/v1

Creates a person patron with a guardian (eg A financial responsible).

Implementation Notes

Returns the id of the patron if the request succeeds. Name and address will be automatically fetched from the CPR-Registry.

If the CPR-Registry is not authorized to provide information about the patron, then response message 404 will be sent back.

If the supplied cpr number of the patron equals that of the guardian, then response message 400 will be sent back.

If the email of the guardian is invalid, then response message 400 will be sent back.

If an email or phone number for the patron is supplied and it is invalid, then response message 400 will be sent back.

In case of a successful creation of the patron, a confirmation email is sent to the guardian. In case of failure an email is sent to guardian stating the creation failed.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronWithGuardianRequest	**The payload with information for the patron to create**	body	Model / Model Schema **PatronWithGuardianRequest {** **cprNumber** (string), **pincode** (string), **phoneNumber** (string, *optional*): Must be valid if supplied, **address** (Address, *optional*),

Parameter	Description	Parameter Type	Data Type
			preferredPickupBranch (string): Must be a valid branch id (eg. DK-761501), **name** (string): The full name of the patron, **guardian** (Guardian), **email** (string, *optional*): Must be valid if supplied **}** **Address {** **country** (string), **city** (string), **street** (string): Street and number, **coName** (string): c/o name, **postalCode** (string) **}** **Guardian {** **cprNumber** (string), **mobilePhoneNumber** (string, *optional*), **address** (Address, *optional*), **name** (string): The full name of the guardian, **email** (string): Must be valid **}**

Response Class

Model | Model Schema

integer

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
200	Ok	
400	bad request	
401	client unauthorized	
404	Data not found	

PUT /external/{agencyid}/patrons/withGuardian/v1 Updates a person patron's guardian (eg A financial responsible).

Implementation Notes

If the person doesn't have a guardian, a new one is created with the information provided. Returns the id of the patron if the request succeeds. Name and address will be automatically fetched from the CPR-Registry.

If the CPR-Registry is not authorized to provide information about the patron and guardian, then response message 404 will be sent back.

If the supplied cpr number of the patron equals that of the guardian, then response message 400 will be sent back.

If the email of the guardian is invalid, then response message 400 will be sent back.

In case of a successful update of the guardian, a confirmation email is sent to the guardian. In case of failure an email is sent to guardian stating the update failed.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
updateGuardianRequest	**The payload with information for the guardian update**	body	Model | Model Schema **UpdateGuardianRequest {** **cprNumber** (string, *optional*): If patronId is provided, this field will be ignored, **patronId** (integer, *optional*), **guardian** (Guardian) **}** **Guardian {** **cprNumber** (string), **mobilePhoneNumber** (string, *optional*), **address** (Address, *optional*), **name** (string): The full name of the guardian, **email** (string): Must be valid **}** **Address {** **country** (string), **city** (string), **street** (string): Street and number,

Parameter	Description	Parameter Type	Data Type
			coName (string): c/o name, **postalCode** (string) }

Response Class

Model | Model Schema

integer

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
200	Ok	
400	bad request	
401	client unauthorized	
404	Data not found	

POST /external/{agencyid}/patrons/withGuardian/v2

Creates a person patron with a guardian (eg A financial responsible).

Implementation Notes

Returns the id of the patron if the request succeeds. Name and address will be automatically fetched from the CPR-Registry. This version includes a possibility of including a BlockStatusRequest in the PatronWithGuardianRequest. The reason of the blockstatusRequest must be one of the following one-letter choices

- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website
- 'E': maximum amount of allowed debt exceeded
- 'D': deceased

If the CPR-Registry is not authorized to provide information about the patron, then response message 404 will be sent back.

If the supplied cpr number of the patron equals that of the guardian, then response message 400 will be sent back.

If the email of the guardian is invalid, then response message 400 will be sent back.

If an email or phone number for the patron is supplied and it is invalid, then response message 400 will be sent back.

If the request includes a blockstatusRequest for the patron, and the blockedSince date is not given or in the wrong format, or the blockedReason is not one of the one-letter choices shown above, then response message 400 will be sent back

In case of a successful creation of the patron, a confirmation email is sent to the guardian. In case of failure an email is sent to guardian stating the creation failed.

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**ISIL of the agency (e.g. DK-761500)**	path	string
`patronWithGuardianRequest`	**The payload with information for the patron to create**	body	Model | Model Schema **PatronWithGuardianRequestV2 {** **cprNumber** (string), **pincode** (string), **phoneNumber** (string, *optional*): Must be valid if supplied, **address** (Address, *optional*), **preferredPickupBranch** (string): Must be a valid branch id (eg. DK-761501), **name** (string): The full name of the patron, **guardian** (Guardian), **blockStatusRequest** (BlockStatusRequest, *optional*), **email** (string, *optional*): Must be valid if supplied **}** **Address {** **country** (string), **city** (string), **street** (string): Street and number, **coName** (string): c/o name, **postalCode** (string)

Parameter	Description	Parameter Type	Data Type
			} **Guardian {** **cprNumber** (string), **mobilePhoneNumber** (string, *optional*), **address** (Address, *optional*), **name** (string): The full name of the guardian, **email** (string): Must be valid } **BlockStatusRequest {** **blockedReason** (string): Reason code for block, **blockedSince** (string): The date from which the patron should be blocked. The dateformat is YYYY-MM-DD, so 9th of March 2023 is written 2023-03-09. }

Response Class

Model | Model Schema

integer

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
200	Ok	
400	bad request	
401	client unauthorized	
404	Data not found	

PUT /external/{agencyid}/patrons/withGuardian/v2 Updates a person patron's guardian (eg A financial responsible).

Implementation Notes

If the person doesn't have a guardian, a new one is created with the information provided. Returns the id of the patron if the request succeeds. Name and address will be automatically fetched from the configured person registry.

If the configured person registry is not authorized to provide information about the patron and guardian, then response message 404 will be sent back.

If the supplied person identifier (e.g. CPR-number) of the patron equals that of the guardian, then response message 400 will be sent back.

If the email of the guardian is invalid, then response message 400 will be sent back.

In case of a successful update of the guardian, a confirmation email is sent to the guardian. In case of failure an email is sent to guardian stating the update failed.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
updateGuardianRequest	**The payload with information for the guardian update**	body	Model | Model Schema **UpdateGuardianRequestV2 {** **patronId** (integer, *optional*), **personId** (string, *optional*): If patronId is provided, this field will be ignored, **guardian** (GuardianV2) **}** **GuardianV2 {** **mobilePhoneNumber** (string, *optional*), **address** (AddressV3, *optional*), **name** (string): The full name of the guardian, **personIdentifier** (string), **email** (string): Must be valid **}** **AddressV3 {** **country** (string): Country, **city** (string): City, **street** (string): Street and number,

Parameter	Description	Parameter Type	Data Type
			coName (string): c/o name, **postalCode** (string): Postal code, **district** (string): Dsitrict, **subDistrict** (string): Subdistrict **}**

Response Class

Model | Model Schema

integer

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
200	Ok	
400	bad request	
401	client unauthorized	
404	Data not found	

POST /external/{agencyid}/patrons/withGuardian/v3 Creates a person patron with a guardian (eg A financial responsible).

Implementation Notes

Returns the id of the patron if the request succeeds. Name and address will be automatically fetched from the configured person registry. This version includes a possibility of including a BlockStatusRequest in the PatronWithGuardianRequest. The reason of the blockstatusRequest must be one of the following one-letter choices

- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website
- 'E': maximum amount of allowed debt exceeded
- 'D': deceased

If the configured person registry is not authorized to provide information about the patron, then response message 404 will be sent back.

If the supplied person identifier (e.g. CPR-number) of the patron equals that of the guardian, then response message 400 will be sent back.

If the email of the guardian is invalid, then response message 400 will be sent back.

If an email or phone number for the patron is supplied and it is invalid, then response message 400 will be sent back.

Multiple email addresses and phone numbers can be stored for a patron.

If multiple email addresses are supplied having receiveNotification as true, then only one of them will be randomly stored as preferred and the rest will be stored as not preferred.

If multiple phone numbers are supplied having receiveNotification as true, then only one of them will be randomly stored as preferred and the rest will be stored as not preferred.

If the request includes a blockstatus for the patron, and the blockedSince date is not given or in the wrong format, or the blockedReason is not one of the one-letter choices shown above, then response message 400 will be sent back

In case of a successful creation of the patron, a confirmation email is sent to the guardian. In case of failure an email is sent to guardian stating the creation failed.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronWithGuardianRequest	**The payload with information for the patron to create**	body	Model Model Schema **PatronWithGuardianRequestV3 {** **pincode** (string), **emailAddresses** (array[EmailAddressV1], *optional*): Must be valid if supplied, **address** (AddressV3, *optional*), **preferredPickupBranch** (string): Must be a valid branch id (eg. DK-761501), **name** (string): The full name of the patron, **personId** (string), **guardian** (GuardianV2), **blockStatusRequest** (BlockStatusRequest, *optional*),

Parameter	Description	Parameter Type	Data Type

```
    phoneNumbers (array[PhoneNumberV1],
    optional): Must be valid if supplied
}
EmailAddressV1 {
    emailAddress (string),
    receiveNotification (boolean)
}
AddressV3 {
    country (string): Country,
    city (string): City,
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string): Postal code,
    district (string): Dsitrict,
    subDistrict (string): Subdistrict
}
GuardianV2 {
    mobilePhoneNumber (string, optional),
    address (AddressV3, optional),
    name (string): The full name of the guardian,
    personIdentifier (string),
    email (string): Must be valid
}
BlockStatusRequest {
    blockedReason (string): Reason code for block,
    blockedSince (string): The date from which the
    patron should be blocked. The dateformat is YYYY-
    MM-DD, so 9th of March 2023 is written 2023-03-
    09.
}
PhoneNumberV1 {
    receiveNotification (boolean),
    phoneNumber (string)
}
```

Response Class

Model | Model Schema

integer

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
200	Ok	
400	bad request	
401	client unauthorized	
404	Data not found	

POST /external/{agencyid}/patrons/withGuardian/v4

Creates a person patron with a guardian (eg A financial responsible).

Implementation Notes

Returns the id of the patron if the request succeeds. Name and address will be automatically fetched from the configured person registry. This version includes a possibility of including a BlockStatusRequest in the PatronWithGuardianRequest. The reason of the blockstatusRequest must be one of the following one-letter choices

- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website
- 'E': maximum amount of allowed debt exceeded
- 'D': deceased

If the configured person registry is not authorized to provide information about the patron, then response message 404 will be sent back.

If the supplied person identifier (e.g. CPR-number) of the patron equals that of the guardian, then response message 400 will be sent back.

If the email of the guardian is invalid, then response message 400 will be sent back.

If an email or phone number for the patron is supplied and it is invalid, then response message 400 will be sent back.

Multiple email addresses and phone numbers can be stored for a patron.

If multiple email addresses are supplied having receiveNotification as true, then only one of them will be randomly stored as preferred and the rest will be stored as not preferred.

If multiple phone numbers are supplied having receiveNotification as true, then only one of them will be randomly stored as preferred and the rest will be stored as not preferred.

If the request includes a blockstatus for the patron, and the blockedSince date is not given or in the wrong format, or the blockedReason is not one of the one-letter choices shown above, then response message 400 will be sent back

In case of a successful creation of the patron, a confirmation email is sent to the guardian. In case of failure an email is sent to guardian stating the creation failed.

Setting nationalRegistryConsent to true will allow the data of the created patron to be synchronized with the Norwegian national registry.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronWithGuardianRequest	**The payload with information for the patron to create**	body	Model Model Schema **PatronWithGuardianRequestV4 {** **pincode** (string, *optional*), **nationalRegistryConsent** (boolean, *optional*), **emailAddresses** (array[EmailAddressV1], *optional*): Must be valid if supplied, **address** (AddressV3, *optional*), **preferredPickupBranch** (string): Must be a valid branch id (eg. DK-761501), **name** (string): The full name of the patron, **personId** (string), **guardian** (GuardianV2), **blockStatusRequest** (BlockStatusRequest, *optional*), **phoneNumbers** (array[PhoneNumberV1], *optional*): Must be valid if supplied **}** **EmailAddressV1 {** **emailAddress** (string), **receiveNotification** (boolean)

Parameter	Description	Parameter Type	Data Type
			} **AddressV3 {** **country** (string): Country, **city** (string): City, **street** (string): Street and number, **coName** (string): c/o name, **postalCode** (string): Postal code, **district** (string): Dsitrict, **subDistrict** (string): Subdistrict } **GuardianV2 {** **mobilePhoneNumber** (string, *optional*), **address** (AddressV3, *optional*), **name** (string): The full name of the guardian, **personIdentifier** (string), **email** (string): Must be valid } **BlockStatusRequest {** **blockedReason** (string): Reason code for block, **blockedSince** (string): The date from which the patron should be blocked. The dateformat is YYYY-MM-DD, so 9th of March 2023 is written 2023-03-09. } **PhoneNumberV1 {** **receiveNotification** (boolean), **phoneNumber** (string) }

Response Class

Model | Model Schema

integer

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
200	Ok	
400	bad request	
401	client unauthorized	
404	Data not found	

Authentication

Show/Hide | List Operations | Expand Operations | Raw

POST /external/v1/{agencyid}/authentication/login Authenticate the client system.

Implementation Notes

The other services can only be used by an authenticated client system. This service authenticates the client system to be able to use the other services.

The response contains a session key which must be supplied in the HTTP header 'X-Session' of all subsequent service calls. The session key can timeout (yielding HTTP Status code 401 on a service call), and if this happens the client system must login again.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency to log into (e.g. DK-761500)**	path	string
login	**credentials for the client system**	body	Model | Model Schema **Login {** **password** (string): Clear text password, **username** (string): Username for the client system **}**

Response Class

Model | Model Schema

ExternalAPIUserInfo {
sessionKey (string): Session key for subsequent API calls

}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
403	invalid client credentials	

Catalog (deprecated version 1)

Show/Hide | List Operations | Expand Operations | Raw

GET /external/v1/{agencyid}/catalog/availability Get availability of bibliographical records.

Implementation Notes

Returns an array of availability for each bibliographical record.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
recordid	**list of record ids**	query	array[string]
exclude	Identifies the branchIds which are excluded from the result	query	array[string]

Response Class

Model | Model Schema

Availability {
  recordId (string): The FAUST number of the Bibliographic record,
  reservable (boolean): True if materials can be reserved,
  available (boolean): True if materials is available on-shelf at some placement, false if all materials are lent out
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/v1/{agencyid}/catalog/bookingInformation/{recordid}

Retrieves all booking information for each branch of an agency for bookings that ends after the current date.

Implementation Notes

Returns an array of BookingBranchInfo which contains:

- the branch ID
- gross number of available materials for that branch
- array of BookingInfo for the given bibliographic record, containing the booking Period and the number of preferred materials

This is to highlight when materials are available for a new potential booking.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
recordid	**identifies the bibliographical record, i.e. the FAUST number**	path	string

Response Class

Model Model Schema

BookingBranchInfo {
branchId (string): The branch ID,
grossNumberAvailable (integer): The gross number of available materials,
bookingInfo (array[BookingInfo]): Details about requested booking information

}
BookingInfo {
preferredMaterials (integer): The preferred number of materials,
period (Period): The booking period information containing the start and the end date
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request com.dantek.dl.rest.RestException	
401	client unauthorized	
404	patron not found	

GET /external/v1/{agencyid}/catalog/holdings Get placement holdings for bibliographical records.

Implementation Notes

Returns an array of holdings for each bibliographical record. The holdings lists the materials on each placement, and whether they are available on-shelf or lent out.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
recordid	**Identifies the bibliographical records - The FAUST number.**	query	array[string]

Parameter	Description	Parameter Type	Data Type
exclude	Identifies the branchIds which are excluded from the result	query	array[string]

Response Class

Model | Model Schema

HoldingsForBibliographicalRecord {
recordId (string): Identifies the bibliographical record for the available materials, The FAUST number,
reservable (boolean): True if there is any reservable materials,
holdings (array[Holdings]): An array of holdings for the materials matching the bibliographical record, as distributed across branches, departments and locations
}
Holdings {
materials (array[Material]): Materials that belongs to this placement,
location (AgencyLocation, *optional*): Placement location,
sublocation (AgencySublocation, *optional*): Placement sublocation,
department (AgencyDepartment, *optional*): Placement department,
branch (AgencyBranch): Placement branch
}
Material {
itemNumber (string): Identifies the material,
periodical (Periodical, *optional*): Present if material is a periodical,
available (boolean): True if material is available on-shelf, false if lent out,
materialGroupName (string): Name of the material group that the material belongs to
}
Periodical {
volume (string, *optional*),
volumeYear (string, *optional*),
displayText (string): A representation of the periodica volume information that is suitable for display,
volumeNumber (string, *optional*)
}
AgencyLocation {
locationId (string): Location identifier,
title (string): Name of the location

}
AgencySublocation {
  title (string): Name of the sub-location,
  sublocationId (string): Sub-location identifier
}
AgencyDepartment {
  departmentId (string): Department identifier,
  title (string): Name of the department
}
AgencyBranch {
  branchId (string): ISIL of branch (e.g. DK-761501),
  title (string): Name of branch
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

Configuration

Show/Hide | List Operations | Expand Operations | Raw

GET /external/v1/{agencyid}/configuration/{key} Get a configuration setting based on a configuration key.

Implementation Notes

Returns a string representation of the setting.

Note: If the settings for the key is a list of values, this method is not suitable for getting the settings.

The list of available configuration settings will be distributed seperately to the clients that needs them, along with a description on how the settings is encoded in the string representation.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
key	**the configuration key to look up**	path	string

Response Class

string

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	key not found	

GET /external/v1/{agencyid}/configuration/{key}/list Get a configuration setting based on a configuration key.

Implementation Notes

Returns an array of strings representation of the setting.

This method is suitable for keys that has a list of values.

The list of available configuration settings will be distributed seperately to the clients that needs them, along with a description on how the settings is encoded in the string representation.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string

Parameter	Description	Parameter Type	Data Type
key	**the configuration key to look up**	path	string

Response Class

Model | Model Schema

array[string]

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	key not found	

Payment (deprecated version 1)

Show/Hide | List Operations | Expand Operations | Raw

GET /external/v1/{agencyid}/patron/{patronid}/fees List of fees in FBS for the patron with all available information about the fee.

Implementation Notes

Returns array of fees.

If the fee covers loaned materials, information about the materials is returned. Each fee in the response includes a 'type', which is used to distinguish between different types of fees.

If the material exists no more, which is the case for fees that are related to closed interlibraryloans, then the fee is still returned, but without material information

The list of available types currently is
fee
compensation
While the type can be used by client systems to look up a suitable display message for the end user, it is important that unrecognized types are treated as 'other'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron that owns the fees**	path	integer
includepaid	true if all paid/unpaid fees should be included, false if only unpaid fees should be included; default=false	query	boolean
includenonpayable	true if fees that are not payable through a CMS system should be included (for read only access); default=false	query	boolean

Response Class

Model | Model Schema

Fee {

payableByClient (boolean): true if the client system is allowed to offer payment for the fee, false if not allowed,
amount (number): The amount to pay, in the currency of the agency,
paidDate (string, *optional*): If the fee has been paid in full, this will be set to the date of the final payment, otherwise not set,
materials (array[FeeMaterial]): Set if fee covers materials,
reasonMessage (string): Human readable free text message about the reason for the fee, presentable to an end user (language is likely to be the mother tongue of the agency),
dueDate (string, *optional*): Expected payment due date,
type (string): Can be used to distinguish between different types of fees,
creationDate (string): The date the fee was created,
feeId (integer): Identifies the fee, used when registering a payment that covers the fee

}

FeeMaterial {

recordId (string): The FAUST number of the bibliographic record,
periodical (Periodical, *optional*): Present if material is a periodical,
materialGroupName (string): Name of the material group that the material belongs to,
materialItemNumber (string): Identifies the exact material covered by the fee

}

Periodical {

volume (string, *optional*),

volumeYear (string, *optional*),
displayText (string): A representation of the periodica volume information that is suitable for display,
volumeNumber (string, *optional*)
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/v1/{agencyid}/patron/{patronid}/payment Pay fees.

Implementation Notes

Returns array of payment confirmations for each fee.

This call is used to inform FBS that a payment have been completed successful from the payment gateway through the CMS client system. The payment contain the order ID from the payment gateway (e.g. dibs) and the fee identifiers for fees covered by the payment. It is expected that a fee has been paid in full when covered by a payment order. The client system is not allowed to offer partial payment of individual fees.

The paymentStatus on the response can be any of these values:
- paymentRegistered
- paymentRegisteredByDifferentOrderId
- paymentNotAllowedByClient

If any other value is encountered, it should be treated as yet another reason for not registerering payment of a fee using the specified order id.

Multiple calls to pay a fee with the same order Id will return the same confirmationId, and the payment will have paymentStatus=='paymentRegistered'.

If a fee has already been paid using a different orderId then no confirmationId is provided, and the payment will have paymentStatus=='paymentRegisteredByDifferentOrderId'.

If the client system was not allowed to offer payment of a fee, then no confirmationId is provided, and the payment will have paymentStatus=='paymentNotAllowedByClient'.

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**ISIL of the agency (e.g. DK-761500)**	path	string
`patronid`	**the patron that owns the fees**	path	integer
`paymentOrder`	**registration of fees covered by a payment order**	body	Model | Model Schema **PaymentOrder {** **orderId** (string): Order Id from payment gateway, **feeIds** (array[integer]): Array of fees fully covered by the order **}**

Response Class

Model | Model Schema

PaymentConfirmation {
orderId (string): Order Id from payment gateway,
confirmationId (string, *optional*): set if fee was registered when using the orderId, unset otherwise (see paymentStatus for reason),
feeId (integer),
paymentStatus (string)
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
`400`	bad request	
`401`	client unauthorized	

Patron Group

Show/Hide | List Operations | Expand Operations | Raw

GET /external/v1/{agencyid}/patrongroups Returns the root group of a specific agency.

Implementation Notes

Returns the root of the PatronGroup and the whole groups hierarchy in the specified agency.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string

Response Class

Model | Model Schema

PatronGroup {
  membersCount (integer): The number of all members in this group, including descendants,
  patronGroupId (integer): The patron group identifier,
  name (string): The name of the group,
  description (string): The description of the group,
  childGroups (array[PatronGroup], *optional*): The array of child groups of this group
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

Booking

Show/Hide | List Operations | Expand Operations | Raw

GET /external/v1/{agencyid}/patrons/{patronid}/bookings Retrieve all bookings made by the patron.

Implementation Notes

Returns an array of booking details in one of the following states

- active
- fulfilled
- beingFulfilled

Bookings having any other state will not be received.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the ID of the patron that owns the bookings**	path	integer

Response Class

Model | Model Schema

Booking {
  recordId (string): The record ID,
  preferredMaterials (integer): The preferred number of materials,
  note (string, *optional*): Additional information about this booking,
  period (Period): The booking period information containing the start and the end date,
  minimumMaterials (integer): The minimum number of materials,
  patronGroupId (integer): The patron group ID for this booking,
  automaticForwardLoan (boolean): True if automatic forward is active for this booking,
  state (string): The booking state,
  requestedFromBranchId (string): The branch that provides the material for booking,
  bookingId (string): The booking identifier,
  deliverBranchId (string): The delivery branch identifier
}
Period {
  from (string, *optional*): Open-ended if not set,
  to (string, *optional*): Open-ended if not set
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request com.dantek.dl.rest.RestException	
401	client unauthorized	
404	patron not found	

POST /external/v1/{agencyid}/patrons/{patronid}/bookings Create a new bookings for the patron.

Implementation Notes

Given a CreateBookingBatch, it creates a list of bookings and returns an array of BookingResponse.

Each response element contains a result of the creation and if ALL results have the value success the created Booking is returned. Otherwise the field is null.

No booking is created if ANY element in the CreateBookingBatch fails to be created

The result of the creation can have the following values:

- success
- notEnoughMaterials
- notUpdatable
- patronDoesNotHavePermission
- other

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the ID of the patron that owns the bookings**	path	integer

batch	**information about bookings that are going to be created**	body	Model Model Schema
			CreateBookingBatch {
			bookings (array[CreateBooking])
			}
			CreateBooking {
			recordId (string): The record ID,

Parameter	Description	Parameter Type	Data Type
			preferredMaterials (integer): The preferred number of materials, **note** (string, *optional*): Additional information about this booking, **period** (Period): The booking period information containing the start and the end date, **minimumMaterials** (integer): The minimum number of materials, **patronGroupId** (integer): The patron group ID for this booking, **automaticForwardLoan** (boolean): True if automatic forward is active for this booking, **requestedFromBranchId** (string): The branch that provides the material for booking, **deliverBranchId** (string): The delivery branch identifier **}** **Period {** **from** (string, *optional*): Open-ended if not set, **to** (string, *optional*): Open-ended if not set **}**

Response Class

Model | Model Schema

BookingResponse {
  result (string): The operation result,
  booking (Booking, *optional*): The booking data as returned by the create/update operation
}
Booking {
  recordId (string): The record ID,
  preferredMaterials (integer): The preferred number of materials,
  note (string, *optional*): Additional information about this booking,
  period (Period): The booking period information containing the start and the end date,
  minimumMaterials (integer): The minimum number of materials,
  patronGroupId (integer): The patron group ID for this booking,
  automaticForwardLoan (boolean): True if automatic forward is active for this booking,
  state (string): The booking state,

requestedFromBranchId (string): The branch that provides the material for booking,
bookingId (string): The booking identifier,
deliverBranchId (string): The delivery branch identifier
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request com.dantek.dl.rest.RestException	
401	client unauthorized	
404	patron not found	

PUT /external/v1/{agencyid}/patrons/{patronid}/bookings

Update existing bookings
Given an UpdateBookingBatch, it updates the list of existing bookings and returns an array of BookingResponse.

Implementation Notes

Each response element contains a result of the update operation and the updated Booking if the operation succeeds. On failure, the result field is updated accordingly and the booking is set to null.

The result of the creation can have the following values:

- success
- notEnoughMaterials
- notUpdatable
- patronDoesNotHavePermission
- other

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid		path	string
patronid	**the ID of the patron that owns the bookings**	path	integer
batch	**information about bookings that are going to be updated**	body	Model | Model Schema **UpdateBookingBatch {** **bookings** (array[Booking]) **}** **Booking {** **recordId** (string): The record ID, **preferredMaterials** (integer): The preferred number of materials, **note** (string, *optional*): Additional information about this booking, **period** (Period): The booking period information containing the start and the end date, **minimumMaterials** (integer): The minimum number of materials, **patronGroupId** (integer): The patron group ID for this booking, **automaticForwardLoan** (boolean): True if automatic forward is active for this booking, **state** (string): The booking state, **requestedFromBranchId** (string): The branch that provides the material for booking, **bookingId** (string): The booking identifier, **deliverBranchId** (string): The delivery branch identifier **}** **Period {** **from** (string, *optional*): Open-ended if not set, **to** (string, *optional*): Open-ended if not set **}**

Response Class

Model | Model Schema

BookingResponse {
result (string): The operation result,
booking (Booking, *optional*): The booking data as returned by the create/update operation
}
Booking {
recordId (string): The record ID,
preferredMaterials (integer): The preferred number of materials,
note (string, *optional*): Additional information about this booking,
period (Period): The booking period information containing the start and the end date,
minimumMaterials (integer): The minimum number of materials,
patronGroupId (integer): The patron group ID for this booking,
automaticForwardLoan (boolean): True if automatic forward is active for this booking,
state (string): The booking state,
requestedFromBranchId (string): The branch that provides the material for booking,
bookingId (string): The booking identifier,
deliverBranchId (string): The delivery branch identifier
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request com.dantek.dl.rest.RestException	
401	client unauthorized	
404	patron not found	

DELETE /external/v1/{agencyid}/patrons/{patronid}/bookings Deletes bookings with the specified IDs.

Implementation Notes

Deletes the bookings corresponding to the given booking IDs

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the ID of the patron that owns the bookings**	path	integer

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request com.dantek.dl.rest.RestException	
401	client unauthorized	
404	patron not found	

Material Loans (deprecated version 1)

Show/Hide | List Operations | Expand Operations | Raw

GET /external/v1/{agencyid}/patrons/{patronid}/loans Get list of current loans by the patron.

Implementation Notes

Returns an array of loans.

If a loan is not renewable then the field renewalStatus will contain a list of one or more of these values:
- deniedReserved
- deniedMaxRenewalsReached
- deniedLoanerIsBlocked
- deniedMaterialIsNotLoanable
- deniedMaterialIsNotFound
- deniedLoanerNotFound
- deniedLoaningProfileNotFound
- deniedOtherReason

If any other value is encountered then it must be treated as 'deniedOtherReason'.

The response contains the field loanDetails.loanType, which can be any of these values:
- loan
- interLibraryLoan
The values are subject to change. If an unrecognized value is encountered, it should be treated as 'other' .

NOTE: Cicero can decide to skip evaluation of the returned loans to minimize response time for loaners with many loans. In that case isRenewable will have the value true, as if it were a successful validation.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron that owns the loans**	path	integer

Response Class

Model | Model Schema

Loan {
isRenewable (boolean): indicates whether this loan can be renewed,
loanDetails (LoanDetails): The loan that was attempted renewed,
isLongtermLoan (boolean): indicates whether this loan is a long term loan,
renewalStatusList (array[string]): if isRenewable == false then this states the reasons for denial
}
LoanDetails {
recordId (string): The FAUST number of the bibliographic record,
loanType (string): The loan type, either **loan** or **interLibraryLoan**,
periodical (Periodical, *optional*): Present if material is a periodical,
dueDate (string): The date when the material must be returned,
ilBibliographicRecord (ILLBibliographicRecord, *optional*): Additional bibliographic information for inter-library loans,
loanDate (string): The date when the material was picked up,
materialGroupName (string): Name of the material group that the material belongs to,
materialItemNumber (string): Identifies the exact material that has been loaned,
loanId (integer): Identifies the loan for use when renewing the loan
}
Periodical {
volume (string, *optional*),

volumeYear (string, *optional*),
displayText (string): A representation of the periodica volume information that is suitable for display,
volumeNumber (string, *optional*)
}
ILLBibliographicRecord {
author (string, *optional*): The author of the material,
isbn (string, *optional*): ISBN-information from the bibliographic record,
periodicalNumber (string, *optional*): Issue number of a periodical,
edition (string, *optional*): Edition-information from the bibliographic record,
language (string, *optional*): Language of the requested material.,
bibliographicCategory (string, *optional*): Bibliographic category from danMARC2 008 *t,
title (string, *optional*): The title of the material,
publicationDateOfComponent (string, *optional*): Publication date of an item component, or article.,
recordId (string): The FAUST number,
issn (string, *optional*): ISSN-information from the bibliographic record,
placeOfPublication (string, *optional*),
mediumType (string): Type of the requested material - from danMARC2 009 *a+*g (general and specific),
periodicalVolume (string, *optional*): Volume name of a periodical,
publisher (string, *optional*): Publisher of the requested material.,
publicationDate (string, *optional*): Publication date of the requested material.
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	patron not found	

GET /external/v1/{agencyid}/patrons/{patronid}/loans/{bookingid} Retrieves material loans for the given booking ID.

Implementation Notes

Retrieves an array of BookingLoan corresponding to the given booking ID.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the ID of the patron that owns the bookings**	path	integer
bookingid	**ID of the booking**	path	string

Response Class

Model | Model Schema

BookingLoan {
patronName (string): The name of the patron that owns the booking,
isRenewable (boolean): indicates whether this loan can be renewed,
loanDetails (LoanDetails): The loan that was attempted renewed,
isLongtermLoan (boolean): indicates whether this loan is a long term loan,
renewalStatusList (array[string]): if isRenewable == false then this states the reasons for denial
}
LoanDetails {
recordId (string): The FAUST number of the bibliographic record,
loanType (string): The loan type, either **loan** or **interLibraryLoan**,
periodical (Periodical, *optional*): Present if material is a periodical,
dueDate (string): The date when the material must be returned,
ilBibliographicRecord (ILLBibliographicRecord, *optional*): Additional bibliographic information for inter-library loans,
loanDate (string): The date when the material was picked up,
materialGroupName (string): Name of the material group that the material belongs to,
materialItemNumber (string): Identifies the exact material that has been loaned,
loanId (integer): Identifies the loan for use when renewing the loan
}
Periodical {
volume (string, *optional*),
volumeYear (string, *optional*),
displayText (string): A representation of the periodica volume information that is suitable for display,

    volumeNumber (string, *optional*)
}
ILLBibliographicRecord {
author (string, *optional*): The author of the material,
isbn (string, *optional*): ISBN-information from the bibliographic record,
periodicalNumber (string, *optional*): Issue number of a periodical,
edition (string, *optional*): Edition-information from the bibliographic record,
language (string, *optional*): Language of the requested material.,
bibliographicCategory (string, *optional*): Bibliographic category from danMARC2 008 *t,
title (string, *optional*): The title of the material,
publicationDateOfComponent (string, *optional*): Publication date of an item component, or article.,
recordId (string): The FAUST number,
issn (string, *optional*): ISSN-information from the bibliographic record,
placeOfPublication (string, *optional*),
mediumType (string): Type of the requested material - from danMARC2 009 *a+*g (general and specific),
periodicalVolume (string, *optional*): Volume name of a periodical,
publisher (string, *optional*): Publisher of the requested material.,
publicationDate (string, *optional*): Publication date of the requested material.
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request com.dantek.dl.rest.RestException	
401	client unauthorized	
404	patron not found	

POST /external/v1/{agencyid}/patrons/{patronid}/loans/renew Renew loans.

Implementation Notes

Returns an array of the updated loans.

If the materials could not be renewed, the return date will be unchanged.

The response field renewalStatus will contain a list of one or more of these values:
- renewed
- deniedReserved
- deniedMaxRenewalsReached
- deniedLoanerIsBlocked
- deniedMaterialIsNotLoanable
- deniedMaterialIsNotFound
- deniedLoanerNotFound
- deniedLoaningProfileNotFound
- deniedOtherReason

If any other value is encountered then it must be treated as 'deniedOtherReason'.

The response contains the field loanDetails.loanType, which can be any of these values:
- loan
- interLibraryLoan

The values are subject to change. If an unrecognized value is encountered, it should be treated as 'other' .

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron that owns the loans**	path	integer
materialLoanIds	**a list of loanId to be renewed**	body	array[integer]

Response Class

Model | Model Schema

RenewedLoan {

loanDetails (LoanDetails): The loan that was attempted renewed,
renewalStatus (array[string]): indicates if renewal was succesful or denied - including the reason for denial.

}

LoanDetails {

recordId (string): The FAUST number of the bibliographic record,
loanType (string): The loan type, either **loan** or **interLibraryLoan**,
periodical (Periodical, *optional*): Present if material is a periodical,
dueDate (string): The date when the material must be returned,
ilBibliographicRecord (ILLBibliographicRecord, *optional*): Additional bibliographic information for inter-library loans,

loanDate (string): The date when the material was picked up,
materialGroupName (string): Name of the material group that the material belongs to,
materialItemNumber (string): Identifies the exact material that has been loaned,
loanId (integer): Identifies the loan for use when renewing the loan
}
Periodical {
volume (string, *optional*),
volumeYear (string, *optional*),
displayText (string): A representation of the periodica volume information that is suitable for display,
volumeNumber (string, *optional*)
}
ILLBibliographicRecord {
author (string, *optional*): The author of the material,
isbn (string, *optional*): ISBN-information from the bibliographic record,
periodicalNumber (string, *optional*): Issue number of a periodical,
edition (string, *optional*): Edition-information from the bibliographic record,
language (string, *optional*): Language of the requested material.,
bibliographicCategory (string, *optional*): Bibliographic category from danMARC2 008 *t,
title (string, *optional*): The title of the material,
publicationDateOfComponent (string, *optional*): Publication date of an item component, or article.,
recordId (string): The FAUST number,
issn (string, *optional*): ISSN-information from the bibliographic record,
placeOfPublication (string, *optional*),
mediumType (string): Type of the requested material - from danMARC2 009 *a+*g (general and specific),
periodicalVolume (string, *optional*): Volume name of a periodical,
publisher (string, *optional*): Publisher of the requested material.,
publicationDate (string, *optional*): Publication date of the requested material.
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

HTTP Status Code	Reason	Response Model
404	patron not found	

Reservations

Show/Hide | List Operations | Expand Operations | Raw

GET /external/v1/{agencyid}/patrons/{patronid}/reservations Get all unfulfilled reservations made by the patron (DEPRECATED).

Implementation Notes

Returns an array of reservation details.

When the patron picks up the reserved materials, the reservation will no longer be returned. Expired or deleted reservations will not be returned.

The response contains reservation state, which can be any of these values:
- reserved
- readyForPickup
- interLibraryReservation
- inTransit
- other

The values are subject to change. If an unrecognized value is encountered, it should be treated as 'other' .

The response contains loanType, which can be any of these values:
- loan
- interLibraryLoan

The values are subject to change. If an unrecognized value is encountered, it should be treated as 'loan' .

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron that owns the reservations**	path	integer

Response Class

Model | Model Schema

ReservationDetails {
recordId (string): The FAUST number,

```
  pickupBranch (string): ISIL-number of pickup branch,
  expiryDate (string): The date when the patron is no longer interested in the reserved material,
  reservationId (integer): Identifies the reservation for use when updating or deleting the reservation,
  pickupDeadline (string, optional): Set if reserved material is available for loan,
  loanType (string),
  dateOfReservation (string),
  periodical (Periodical, optional): Present if material is a periodical,
  ilBibliographicRecord (ILLBibliographicRecord, optional): Additional bibliographic information for inter-library loans,
  state (string),
  numberInQueue (integer, optional): The number in the reservation queue.,
  pickupNumber (string, optional): The reservation number. Will be present if the reservation is ready for pickup (the state is 'readyForPickup')
}
Periodical {
  volume (string, optional),
  volumeYear (string, optional),
  displayText (string): A representation of the periodica volume information that is suitable for display,
  volumeNumber (string, optional)
}
ILLBibliographicRecord {
  author (string, optional): The author of the material,
  isbn (string, optional): ISBN-information from the bibliographic record,
  periodicalNumber (string, optional): Issue number of a periodical,
  edition (string, optional): Edition-information from the bibliographic record,
  language (string, optional): Language of the requested material.,
  bibliographicCategory (string, optional): Bibliographic category from danMARC2 008 *t,
  title (string, optional): The title of the material,
  publicationDateOfComponent (string, optional): Publication date of an item component, or article.,
  recordId (string): The FAUST number,
  issn (string, optional): ISSN-information from the bibliographic record,
  placeOfPublication (string, optional),
  mediumType (string): Type of the requested material - from danMARC2 009 *a+*g (general and specific),
  periodicalVolume (string, optional): Volume name of a periodical,
  publisher (string, optional): Publisher of the requested material.,
  publicationDate (string, optional): Publication date of the requested material.
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
`400`	bad request	
`401`	client unauthorized	

POST **/external/v1/{agencyid}/patrons/{patronid}/reservations** Create new reservations for the patron (DEPRECATED).

Implementation Notes

Returns an array of reservation details for the created reservations.

The response contains reservation state, which can be any of these values:
- reserved
- readyForPickup
- interLibraryReservation
- inTransit
- other

The values are subject to change. If an unrecognized value is encountered, it should be treated as 'other'.

The response contains loanType, which can be any of these values:
- loan
- interLibraryLoan

The values are subject to change. If an unrecognized value is encountered, it should be treated as 'other' .

When making a reservation of a periodical, the values to put in the PeriodicalReservation structure can be obtained from the periodical information retrieved with the Catalog service.

This method has been deprecated use /external/v1/{agencyid}/patrons/{patronid}/reservations/v2 instead

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**ISIL of the agency (e.g. DK-761500)**	path	string
`patronid`	**the patron that makes the reservations**	path	integer
`createReservationBatch`	**the reservations to be created**	body	

Parameter	Description	Parameter Type	Data Type
			Model Model Schema **CreateReservationBatch {** **reservations** (array[CreateReservation]): Reservations with duplicate record id's will be removed. **}** **CreateReservation {** **recordId** (string): Identifies the bibliographical record to reserve - The FAUST number, **expiryDate** (string, *optional*): The date where the patron is no longer interested in the reserved material. If not set, a date will be calculated from the agency default interest period, **pickupBranch** (string, *optional*): ISIL-number of pickup branch. If not set, will default to patrons preferred pickup branch, **periodical** (PeriodicalReservation, *optional*): Present if making reservation on a periodical **}** **PeriodicalReservation {** **volume** (string, *optional*), **volumeYear** (string, *optional*), **volumeNumber** (string, *optional*) **}**

Response Class

Model Model Schema

ReservationDetails {
recordId (string): The FAUST number,
pickupBranch (string): ISIL-number of pickup branch,
expiryDate (string): The date when the patron is no longer interested in the reserved material,
reservationId (integer): Identifies the reservation for use when updating or deleting the reservation,
pickupDeadline (string, *optional*): Set if reserved material is available for loan,
loanType (string),
dateOfReservation (string),

```
    periodical (Periodical, optional): Present if material is a periodical,
    ilBibliographicRecord (ILLBibliographicRecord, optional): Additional bibliographic information for inter-library loans,
    state (string),
    numberInQueue (integer, optional): The number in the reservation queue.,
    pickupNumber (string, optional): The reservation number. Will be present if the reservation is ready for pickup (the state is 'readyForPickup')
}
Periodical {
    volume (string, optional),
    volumeYear (string, optional),
    displayText (string): A representation of the periodica volume information that is suitable for display,
    volumeNumber (string, optional)
}
ILLBibliographicRecord {
    author (string, optional): The author of the material,
    isbn (string, optional): ISBN-information from the bibliographic record,
    periodicalNumber (string, optional): Issue number of a periodical,
    edition (string, optional): Edition-information from the bibliographic record,
    language (string, optional): Language of the requested material.,
    bibliographicCategory (string, optional): Bibliographic category from danMARC2 008 *t,
    title (string, optional): The title of the material,
    publicationDateOfComponent (string, optional): Publication date of an item component, or article.,
    recordId (string): The FAUST number,
    issn (string, optional): ISSN-information from the bibliographic record,
    placeOfPublication (string, optional),
    mediumType (string): Type of the requested material - from danMARC2 009 *a+*g (general and specific),
    periodicalVolume (string, optional): Volume name of a periodical,
    publisher (string, optional): Publisher of the requested material.,
    publicationDate (string, optional): Publication date of the requested material.
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	

HTTP Status Code	Reason	Response Model
401	client unauthorized	
404	patron not found	

PUT /external/v1/{agencyid}/patrons/{patronid}/reservations — Update existing reservations.

Implementation Notes

Returns an array of the updated reservation details.

The response contains reservation state, which can be any of these values:
- reserved
- readyForPickup
- interLibraryReservation
- inTransit
- other

The values are subject to change. If an unrecognized value is encountered, it should be treated as 'other'.

The response contains loanType, which can be any of these values:
- loan
- interLibraryLoan

The values are subject to change. If an unrecognized value is encountered, it should be treated as 'other' .

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron that owns the reservations**	path	integer

reservations	**the reservations to be updated**	body	Model / Model Schema
			UpdateReservationBatch {
			reservations (array[UpdateReservation])
			}
			UpdateReservation {
			expiryDate (string, *optional*): The date where the patron is no longer interested in the reserved material. If not set,

Parameter	Description	Parameter Type	Data Type
			the reservation will keep the original date., **pickupBranch** (string, *optional*): ISIL-number of pickup branch. If not set, the reservation will keep the original pickup branch., **reservationId** (integer): Identifies the reservation }

Response Class

Model | Model Schema

ReservationDetails {
recordId (string): The FAUST number,
pickupBranch (string): ISIL-number of pickup branch,
expiryDate (string): The date when the patron is no longer interested in the reserved material,
reservationId (integer): Identifies the reservation for use when updating or deleting the reservation,
pickupDeadline (string, *optional*): Set if reserved material is available for loan,
loanType (string),
dateOfReservation (string),
periodical (Periodical, *optional*): Present if material is a periodical,
ilBibliographicRecord (ILLBibliographicRecord, *optional*): Additional bibliographic information for inter-library loans,
state (string),
numberInQueue (integer, *optional*): The number in the reservation queue.,
pickupNumber (string, *optional*): The reservation number. Will be present if the reservation is ready for pickup (the state is 'readyForPickup')
}
Periodical {
volume (string, *optional*),
volumeYear (string, *optional*),
displayText (string): A representation of the periodica volume information that is suitable for display,
volumeNumber (string, *optional*)
}
ILLBibliographicRecord {
author (string, *optional*): The author of the material,
isbn (string, *optional*): ISBN-information from the bibliographic record,
periodicalNumber (string, *optional*): Issue number of a periodical,
edition (string, *optional*): Edition-information from the bibliographic record,

language (string, *optional*): Language of the requested material.,
bibliographicCategory (string, *optional*): Bibliographic category from danMARC2 008 *t,
title (string, *optional*): The title of the material,
publicationDateOfComponent (string, *optional*): Publication date of an item component, or article.,
recordId (string): The FAUST number,
issn (string, *optional*): ISSN-information from the bibliographic record,
placeOfPublication (string, *optional*),
mediumType (string): Type of the requested material - from danMARC2 009 *a+*g (general and specific),
periodicalVolume (string, *optional*): Volume name of a periodical,
publisher (string, *optional*): Publisher of the requested material.,
publicationDate (string, *optional*): Publication date of the requested material.
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

DELETE /external/v1/{agencyid}/patrons/{patronid}/reservations

Delete existing reservations.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron that owns the reservations**	path	integer
reservationid	**a list of reservation ids for reservations that are to be deleted**	query	array[integer]

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/v1/{agencyid}/patrons/{patronid}/reservations/add Create new reservations for the patron (DEPRECATED).

Implementation Notes

Given a CreateReservationBatch, it creates a list of reservations and returns a ReservationResponse.

ReservationResponse.success indicates if the reservations were created sucessfully. If any of the reservations have failed then all reservations will be failed and ReservationResponse.success will be false. If all reservations are successfully created ReservationResponse.success will be true.

ReservationResponse.reservationResults contains details about each reservation. A ReservationResult.result has the status of a reservation and can be any of the following values:

- success
- patron_is_blocked
- patron_not_found
- already_reserved
- already_loaned
- material_not_loanable
- material_not_reservable
- material_lost
- material_Discarded
- loaning_profile_not_found
- material_not_found
- material_part_of_collection
- not_reservable
- no_reservable_materials
- interlibrary_material_not_reservable
- previously_loaned_by_homebound_patron
- exceeds_max_reservations

The values are subject to change. If an unrecognized value is encountered, it should be treated as an error.

The reservation detail in the response contains a reservation state, which can be any of these values:
- reserved
- readyForPickup
- interLibraryReservation
- inTransit
- other

The values are subject to change. If an unrecognized value is encountered, it should be treated as 'other'.

The reservation detail contains loanType, which can be any of these values:
- loan
- interLibraryLoan
The values are subject to change.

When making a reservation of a periodical, the values to put in the PeriodicalReservation structure can be obtained from the periodical information retrieved with the Catalog service.

This method has been deprecated use /external/v1/{agencyid}/patrons/{patronid}/reservations/v2 instead

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**ISIL of the agency (e.g. DK-761500)**	path	string
`patronid`	**the patron that makes the reservations**	path	integer
`createReservationBatch`	**the reservations to be created**	body	Model Model Schema **CreateReservationBatch {** **reservations** (array[CreateReservation]): Reservations with duplicate record id's will be removed. **}** **CreateReservation {** **recordId** (string): Identifies the bibliographical record to reserve - The FAUST number, **expiryDate** (string, *optional*): The date where the patron is no longer interested in the reserved material. If not set, a date will be calculated from the agency default interest period, **pickupBranch** (string, *optional*): ISIL-number of pickup branch. If not set, will default to patrons preferred pickup branch, **periodical** (PeriodicalReservation, *optional*): Present if making reservation on a periodical **}** **PeriodicalReservation {**

Parameter	Description	Parameter Type	Data Type
			volume (string, *optional*), **volumeYear** (string, *optional*), **volumeNumber** (string, *optional*) **}**

Response Class

Model | Model Schema

ReservationResponse {
  success (boolean): True if all reservation were create successfully otherwise false,
  reservationResults (array[ReservationResult]): Result of each reservation
}
ReservationResult {
  recordId (string): Recordid of the record to reserve,
  result (string): The reservation result,
  periodical (PeriodicalReservation, *optional*): Periodical information of the reservation,
  reservationDetails (ReservationDetails, *optional*): The reservation data as returned by the create/update operation
}
PeriodicalReservation {
  volume (string, *optional*),
  volumeYear (string, *optional*),
  volumeNumber (string, *optional*)
}
ReservationDetails {
  recordId (string): The FAUST number,
  pickupBranch (string): ISIL-number of pickup branch,
  expiryDate (string): The date when the patron is no longer interested in the reserved material,
  reservationId (integer): Identifies the reservation for use when updating or deleting the reservation,
  pickupDeadline (string, *optional*): Set if reserved material is available for loan,
  loanType (string),
  dateOfReservation (string),
  periodical (Periodical, *optional*): Present if material is a periodical,
  ilBibliographicRecord (ILLBibliographicRecord, *optional*): Additional bibliographic information for inter-library loans,
  state (string),
  numberInQueue (integer, *optional*): The number in the reservation queue.,

pickupNumber (string, *optional*): The reservation number. Will be present if the reservation is ready for pickup (the state is 'readyForPickup')
}
Periodical {
volume (string, *optional*),
volumeYear (string, *optional*),
displayText (string): A representation of the periodica volume information that is suitable for display,
volumeNumber (string, *optional*)
}
ILLBibliographicRecord {
author (string, *optional*): The author of the material,
isbn (string, *optional*): ISBN-information from the bibliographic record,
periodicalNumber (string, *optional*): Issue number of a periodical,
edition (string, *optional*): Edition-information from the bibliographic record,
language (string, *optional*): Language of the requested material.,
bibliographicCategory (string, *optional*): Bibliographic category from danMARC2 008 *t,
title (string, *optional*): The title of the material,
publicationDateOfComponent (string, *optional*): Publication date of an item component, or article.,
recordId (string): The FAUST number,
issn (string, *optional*): ISSN-information from the bibliographic record,
placeOfPublication (string, *optional*),
mediumType (string): Type of the requested material - from danMARC2 009 *a+*g (general and specific),
periodicalVolume (string, *optional*): Volume name of a periodical,
publisher (string, *optional*): Publisher of the requested material.,
publicationDate (string, *optional*): Publication date of the requested material.
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	patron not found	

GET /external/v1/{agencyid}/patrons/{patronid}/reservations/v2 Get all unfulfilled reservations made by the patron.

Implementation Notes

Returns an array of reservation details.

When the patron picks up the reserved materials, the reservation will no longer be returned. Expired or deleted reservations will not be returned.

The response contains reservation state, which can be any of these values:
- reserved
- readyForPickup
- interLibraryReservation
- inTransit
- other

The values are subject to change. If an unrecognized value is encountered, it should be treated as 'other' .

The response contains reservationType, which can be any of these values:
- NORMAL
- PARALLEL
- SERIAL
- INTER_LIBRARY

The values are subject to change. If an unrecognized value is encountered, iit should be treated as 'normal'

The response contains a transactionId, which links together parallel reservations.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron that owns the reservations**	path	integer

Response Class

Model | Model Schema

ReservationDetailsV2 {
pickupBranch (string): ISIL-number of pickup branch,
pickupDeadline (string, *optional*): Set if reserved material is available for loan,
dateOfReservation (string),
ilBibliographicRecord (ILLBibliographicRecord, *optional*): Additional bibliographic information for inter-library loans,

```
    numberInQueue (integer, optional): The number in the reservation queue.,
    pickupNumber (string, optional): The reservation number. Will be present if the reservation is ready for pickup (the state is 'readyForPickup'),
    transactionId (string): Identifies the transaction of reservations,
    recordId (string): The FAUST number,
    expiryDate (string): The date when the patron is no longer interested in the reserved material,
    reservationId (integer): Identifies the reservation for use when updating or deleting the reservation,
    periodical (Periodical, optional): Present if material is a periodical,
    reservationType (string),
    state (string)
}
ILLBibliographicRecord {
    author (string, optional): The author of the material,
    isbn (string, optional): ISBN-information from the bibliographic record,
    periodicalNumber (string, optional): Issue number of a periodical,
    edition (string, optional): Edition-information from the bibliographic record,
    language (string, optional): Language of the requested material.,
    bibliographicCategory (string, optional): Bibliographic category from danMARC2 008 *t,
    title (string, optional): The title of the material,
    publicationDateOfComponent (string, optional): Publication date of an item component, or article.,
    recordId (string): The FAUST number,
    issn (string, optional): ISSN-information from the bibliographic record,
    placeOfPublication (string, optional),
    mediumType (string): Type of the requested material - from danMARC2 009 *a+*g (general and specific),
    periodicalVolume (string, optional): Volume name of a periodical,
    publisher (string, optional): Publisher of the requested material.,
    publicationDate (string, optional): Publication date of the requested material.
}
Periodical {
    volume (string, optional),
    volumeYear (string, optional),
    displayText (string): A representation of the periodica volume information that is suitable for display,
    volumeNumber (string, optional)
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/v1/{agencyid}/patrons/{patronid}/reservations/v2 Create new reservations for the patron.

Implementation Notes

Given a CreateReservationBatch, it creates a list of reservations and returns a ReservationResponse.

The CreateReservationBatch.type indicates the reservation type of the request. If left out the request will be considered of type normal. The type can be any of the following values:

- normal
- parallel
The values are subject to change.

ReservationResponse.success indicates if the reservations were created sucessfully. If any of the reservations have failed then all reservations will be failed and ReservationResponse.success will be false. If all reservations are successfully created ReservationResponse.success will be true.

ReservationResponse.reservationResults contains details about each reservation. A ReservationResult.result has the status of a reservation and can be any of the following values:

- success
- patron_is_blocked
- patron_not_found
- already_reserved
- already_loaned
- material_not_loanable
- material_not_reservable
- material_lost
- material_Discarded
- loaning_profile_not_found
- material_not_found
- material_part_of_collection
- not_reservable
- no_reservable_materials
- interlibrary_material_not_reservable
- previously_loaned_by_homebound_patron

- exceeds_max_reservations

The values are subject to change. If an unrecognized value is encountered, it should be treated as an error.

The reservation detail in the response contains a reservation state, which can be any of these values:
- reserved
- readyForPickup
- interLibraryReservation
- inTransit
- other

The values are subject to change. If an unrecognized value is encountered, it should be treated as 'other'.

When making a reservation of a periodical, the values to put in the PeriodicalReservation structure can be obtained from the periodical information retrieved with the Catalog service.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron that makes the reservations**	path	integer
createReservationBatch	**the reservations to be created**	body	Model | Model Schema **CreateReservationBatchV2 {** **reservations** (array[CreateReservation]): Reservations with duplicate record id's will be removed., **type** (string, *optional*): Will be considered normal if not set **}** **CreateReservation {** **recordId** (string): Identifies the bibliographical record to reserve - The FAUST number, **expiryDate** (string, *optional*): The date where the patron is no longer interested in the reserved material. If not set, a date will be calculated from the agency default interest period, **pickupBranch** (string, *optional*): ISIL-number of pickup branch. If not set, will default to patrons preferred pickup branch,

Parameter	Description	Parameter Type	Data Type
			periodical (PeriodicalReservation, *optional*): Present if making reservation on a periodical **}** **PeriodicalReservation {** **volume** (string, *optional*), **volumeYear** (string, *optional*), **volumeNumber** (string, *optional*) **}**

Response Class

Model | Model Schema

ReservationResponseV2 {
  success (boolean): True if all reservations were created successfully otherwise false,
  reservationResults (array[ReservationResultV2]): Result of each reservation
}
ReservationResultV2 {
  recordId (string): Recordid of the record to reserve,
  result (string): The reservation result,
  periodical (PeriodicalReservation, *optional*): Periodical information of the reservation,
  reservationDetails (ReservationDetailsV2, *optional*): The reservation data as returned by the create/update operation
}
PeriodicalReservation {
  volume (string, *optional*),
  volumeYear (string, *optional*),
  volumeNumber (string, *optional*)
}
ReservationDetailsV2 {
  pickupBranch (string): ISIL-number of pickup branch,
  pickupDeadline (string, *optional*): Set if reserved material is available for loan,
  dateOfReservation (string),
  ilBibliographicRecord (ILLBibliographicRecord, *optional*): Additional bibliographic information for inter-library loans,
  numberInQueue (integer, *optional*): The number in the reservation queue.,
  pickupNumber (string, *optional*): The reservation number. Will be present if the reservation is ready for pickup (the state is 'readyForPickup'),
  transactionId (string): Identifies the transaction of reservations,

recordId (string): The FAUST number,
expiryDate (string): The date when the patron is no longer interested in the reserved material,
reservationId (integer): Identifies the reservation for use when updating or deleting the reservation,
periodical (Periodical, *optional*): Present if material is a periodical,
reservationType (string),
state (string)
}
ILLBibliographicRecord {
author (string, *optional*): The author of the material,
isbn (string, *optional*): ISBN-information from the bibliographic record,
periodicalNumber (string, *optional*): Issue number of a periodical,
edition (string, *optional*): Edition-information from the bibliographic record,
language (string, *optional*): Language of the requested material.,
bibliographicCategory (string, *optional*): Bibliographic category from danMARC2 008 *t,
title (string, *optional*): The title of the material,
publicationDateOfComponent (string, *optional*): Publication date of an item component, or article.,
recordId (string): The FAUST number,
issn (string, *optional*): ISSN-information from the bibliographic record,
placeOfPublication (string, *optional*),
mediumType (string): Type of the requested material - from danMARC2 009 *a+*g (general and specific),
periodicalVolume (string, *optional*): Volume name of a periodical,
publisher (string, *optional*): Publisher of the requested material.,
publicationDate (string, *optional*): Publication date of the requested material.
}
Periodical {
volume (string, *optional*),
volumeYear (string, *optional*),
displayText (string): A representation of the periodica volume information that is suitable for display,
volumeNumber (string, *optional*)
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	patron not found	

Patron (deprecated version 1)

Show/Hide | List Operations | Expand Operations | Raw

POST /external/v1/{agencyid}/patrons Create a new patron who is a person.

Implementation Notes

When a patron doesn't have a patron account in the library system, but logs in using a trusted authentication source (e.g NemId), the patron account can be created using this service. Name and address will be automatically fetched from CPR-Registry, and cannot be supplied by the client. If the CPR-Registry is not authorized to provide information about the patron, then repsonse message 404 will be sent back

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
createPatronRequest	**the patron to be created**	body	Model Model Schema **CreatePatronRequest {** **cprNumber** (string), **pincode** (string), **patron** (PatronSettings)

Parameter	Description	Parameter Type	Data Type
			} **PatronSettings {** **emailAddress** (string, *optional*): Required if patron should receive email notifications Existing email addresses are overwritten with this value If left empty existing email addresses are deleted, **phoneNumber** (string, *optional*): Required if patron should receive SMS notifications Existing phonenumbers are overwritten with this value If left empty existing phonenumbers are deleted, **preferredPickupBranch** (string): ISIL-number of preferred pickup branch, **onHold** (Period, *optional*): If not set then the patron is not on hold, **receiveEmail** (boolean), **receivePostalMail** (boolean): This field is deprecated and is no longer used, **receiveSms** (boolean) } **Period {** **from** (string, *optional*): Open-ended if not set, **to** (string, *optional*): Open-ended if not set }

Response Class

Model | Model Schema

AuthenticatedPatron {
authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,
patron (Patron, *optional*): Only available if patron exists in FBS and was succesfully authenticated.
}
Patron {
birthday (string, *optional*),
coAddress (Address, *optional*),
address (Address, *optional*),
preferredPickupBranch (string): ISIL of preferred pickup branch,

```
    onHold (Period, optional): If not set then the patron is not on hold,
    patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
    receiveEmail (boolean),
    blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
    receiveSms (boolean),
    emailAddress (string, optional),
    phoneNumber (string, optional),
    name (string, optional),
    receivePostalMail (boolean): This field is deprecated and is no longer used,
    allowBookings (boolean, optional): True if the user is allowed to create bookings.,
    defaultInterestPeriod (integer): Length of default interest period in days,
    resident (boolean): True if the user is resident in the same municipality as the library
}
Address {
    country (string),
    city (string),
    street (string): Street and number,
    postalCode (string)
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	

HTTP Status Code	Reason	Response Model
401	client unauthorized	
404	Data not found	

PUT /external/v1/{agencyid}/patrons/{patronid} Update information about the patron.

Implementation Notes

The name and address cannot be supplied by the client. If the CPR-Registry is not authorized to provide information about the patron, then the name and address will not be updated.

It is possible to either update just the pincode, update just some patron settings, or update both.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron to be updated**	path	integer

updatePatron	**updated information about the patron**	body	Model Model Schema
			UpdatePatronRequest {
			patron (PatronSettings, *optional*): Set this if patron details are to be changed,
			pincodeChange (PincodeChange, *optional*): Set this if pincode is to be changed
			}
			PatronSettings {

Parameter	Description	Parameter Type	Data Type
			emailAddress (string, *optional*): Required if patron should receive email notifications Existing email addresses are overwritten with this value If left empty existing email addresses are deleted, **phoneNumber** (string, *optional*): Required if patron should receive SMS notifications Existing phonenumbers are overwritten with this value If left empty existing phonenumbers are deleted, **preferredPickupBranch** (string): ISIL-number of preferred pickup branch, **onHold** (Period, *optional*): If not set then the patron is not on hold, **receiveEmail** (boolean), **receivePostalMail** (boolean): This field is deprecated and is no longer used, **receiveSms** (boolean) **}** **Period {** **from** (string, *optional*): Open-ended if not set, **to** (string, *optional*): Open-ended if not set **}** **PincodeChange {** **pincode** (string): The new pincode for the libraryCard, **libraryCardNumber** (string): Identifies the libraryCard for which the pincode is to be changed. This can be either a physical card or the CPR number, that is used as a libraryCard **}**

Response Class

Model | Model Schema

AuthenticatedPatron {

authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,

patron (Patron, *optional*): Only available if patron exists in FBS and was succesfully authenticated.

```
}
Patron {
   birthday (string, optional),
   coAddress (Address, optional),
   address (Address, optional),
   preferredPickupBranch (string): ISIL of preferred pickup branch,
   onHold (Period, optional): If not set then the patron is not on hold,
   patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
   receiveEmail (boolean),
   blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
   receiveSms (boolean),
   emailAddress (string, optional),
   phoneNumber (string, optional),
   name (string, optional),
   receivePostalMail (boolean): This field is deprecated and is no longer used,
   allowBookings (boolean, optional): True if the user is allowed to create bookings.,
   defaultInterestPeriod (integer): Length of default interest period in days,
   resident (boolean): True if the user is resident in the same municipality as the library
}
Address {
   country (string),
   city (string),
   street (string): Street and number,
   postalCode (string)
}
Period {
   from (string, optional): Open-ended if not set,
   to (string, optional): Open-ended if not set
}
BlockStatus {
   blockedReason (string): Reason code for block,
   blockedSince (string),
   message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	patron not found	

POST /external/v1/{agencyid}/patrons/authenticate

Authenticates a patron and returns the patron details.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string

Parameter	Description	Parameter Type	Data Type
`authenticationRequest`	**credentials for patron to be authenticated**	body	Model | Model Schema **AuthenticationRequest {** **pincode** (string): The pincode that belongs to the libraryCardNumber in plain text, **libraryCardNumber** (string): Identifies a libraryCard. This can be either a physical card or a CPR number that is used as a libraryCard **}**

Response Class

Model | Model Schema

AuthenticatedPatron {

authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,

patron (Patron, *optional*): Only available if patron exists in FBS and was succesfully authenticated.

}

Patron {

birthday (string, *optional*),
coAddress (Address, *optional*),
address (Address, *optional*),
preferredPickupBranch (string): ISIL of preferred pickup branch,
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean): This field is deprecated and is no longer used,
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library

}
Address {
  country (string),
  city (string),
  street (string): Street and number,
  postalCode (string)
}
Period {
  from (string, *optional*): Open-ended if not set,
  to (string, *optional*): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/v1/{agencyid}/patrons/preauthenticated

Returns the patron details of a patron that the client has pre-authenticated using a third party.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron. Note: Whis method can only be used for patrons who are people, and not e.g. Companies or Libraries.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen

- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**ISIL of the agency (e.g. DK-761500)**	path	string
`cprNumber`	**CPR-number of the patron**	body	string

Response Class

Model | Model Schema

AuthenticatedPatron {
authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,
patron (Patron, *optional*): Only available if patron exists in FBS and was succesfully authenticated.
}
Patron {
birthday (string, *optional*),
coAddress (Address, *optional*),
address (Address, *optional*),
preferredPickupBranch (string): ISIL of preferred pickup branch,
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean): This field is deprecated and is no longer used,
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,

```
    defaultInterestPeriod (integer): Length of default interest period in days,
    resident (boolean): True if the user is resident in the same municipality as the library
}
Address {
    country (string),
    city (string),
    street (string): Street and number,
    postalCode (string)
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/v1/{agencyid}/patrons/preauthenticated/unic

Returns the patron details of a patron that the client has pre-authenticated using UNIC.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron. Note: Whis method can only be used for patrons who are persons, and not e.g. Companies or Libraries.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
unicUsername	**UNIC username of the patron**	body	string

Response Class

Model | Model Schema

AuthenticatedPatron {

authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,
patron (Patron, *optional*): Only available if patron exists in FBS and was succesfully authenticated.

}

Patron {

birthday (string, *optional*),
coAddress (Address, *optional*),
address (Address, *optional*),
preferredPickupBranch (string): ISIL of preferred pickup branch,
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean): This field is deprecated and is no longer used,

allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library
}
Address {
country (string),
city (string),
street (string): Street and number,
postalCode (string)
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
BlockStatus {
blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

Placement

Show/Hide | List Operations | Expand Operations | Raw

GET /external/v1/{agencyid}/{branchId}/closingdates Get closing dates for a branch in an interval, inclusively between startDate and endDate.

Implementation Notes

Returns array of closing dates.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
branchId	**ISIL of branch (e.g. DK-761501)**	path	string
startDate	Starting date of the interval in the format yyyy-mm-dd	query	string
endDate	End date of the interval in the format yyyy-mm-dd	query	string

Response Class

Model | Model Schema

```
ClosingDateV1 {
  branchId (string),
  description (string),
  closingDate (string)
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/v1/{agencyid}/{branchId}/openinghours Get opening hours for a branch.

Implementation Notes

Returns array of opening hours.

Parameters

Parameter	Description	Parameter Type	Data Type
`agencyid`	**ISIL of the agency (e.g. DK-761500)**	path	string
`branchId`	**ISIL of branch (e.g. DK-761501)**	path	string

Response Class

Model | Model Schema

OpeningHoursV1 {
branchId (string): ISIL of branch (e.g. DK-761501),
serviceHoursStartTime (string): The opening time for the service hours Format: HH:MM:SS,
serviceHoursEndTime (string): The closing time for the service hours Format: HH:MM:SS,
openHoursEndTime (string): The closing time for the opening hours Format: HH:MM:SS,
day (string) = ['MONDAY' or 'TUESDAY' or 'WEDNESDAY' or 'THURSDAY' or 'FRIDAY' or 'SATURDAY' or 'SUNDAY']: The day of the specified opening hours,
openHoursStartTime (string): The opening time for the opening hours. Format: HH:MM:SS
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
`400`	bad request	
`401`	client unauthorized	

GET /external/v1/{agencyid}/branches Get branches for an agency.

Implementation Notes

Returns array of branches.

Can be used for giving the patron the option of choosing a preferred branch or where to pick up reservations.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
exclude	Identifies the branchIds which are excluded from the result	query	array[string]

Response Class

Model | Model Schema

AgencyBranch {
  branchId (string): ISIL of branch (e.g. DK-761501),
  title (string): Name of branch
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/v1/{agencyid}/branches/contact Get contact information for branches.

Implementation Notes

Returns a list of branch contacts.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
branchIds	List of branch IDs to get contact information for	query	array[string]

Response Class

Model | Model Schema

BranchContact {
- **branchId** (string): ISIL of the branch (e.g. DK-761501),
- **zipCode** (string): ZIP code of the branch,
- **website** (string): Website of the branch,
- **address** (string): Address of the branch,
- **city** (string): City where the branch is located,
- **phone** (string): Phone number of the branch,
- **title** (string): Name of the branch,
- **email** (string): Email address of the branch

}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/v1/{agencyid}/departments

Get translations from department identifiers to displayable text.

Implementation Notes

Returns array of departments.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string

Response Class

Model | Model Schema

AgencyDepartment {
  departmentId (string): Department identifier,
  title (string): Name of the department
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/v1/{agencyid}/locations

Get translations from location identifiers to displayable text.

Implementation Notes

Returns array of locations.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string

Response Class

Model | Model Schema

AgencyLocation {
  locationId (string): Location identifier,
  title (string): Name of the location
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/v1/{agencyid}/sections Get translations from section identifiers to displayable text.

Implementation Notes

Returns array of sections.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string

Response Class

Model | Model Schema

AgencySection {
sectionId (string): Section identifier,
title (string): Name of the section
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/v1/{agencyid}/sublocations Get translations from sub-location identifiers to displayable text.

Implementation Notes

Returns array ofsub-locations.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string

Response Class

Model | Model Schema

AgencySublocation {
title (string): Name of the sub-location,
sublocationId (string): Sub-location identifier
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

Catalog (deprecated version 2)

Show/Hide | List Operations | Expand Operations | Raw

GET /external/v2/{agencyid}/catalog/availability Get availability of bibliographical records.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string

Parameter	Description	Parameter Type	Data Type
recordid	**list of record ids**	query	array[string]
exclude	Identifies the branchIds which are excluded from the result	query	array[string]

Response Class

Model | Model Schema

Availability {
  recordId (string): The FAUST number of the Bibliographic record,
  reservations (integer): Total number of current active reservations of the Bibliographic record,
  reservable (boolean): True if materials can be reserved,
  available (boolean): True if materials is available on-shelf at some placement, false if all materials are lent out
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

GET /external/v2/{agencyid}/catalog/holdings

Get placement holdings for bibliographical records.

Implementation Notes

Returns an array of holdings for each bibliographical record together with the total number of current active reservations. The holdings lists the materials on each placement, and whether they are available on-shelf or lent out.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string

Parameter	Description	Parameter Type	Data Type
`recordid`	**Identifies the bibliographical records - The FAUST number.**	query	array[string]
`exclude`	Identifies the branchIds which are excluded from the result	query	array[string]

Response Class

Model | Model Schema

HoldingsForBibliographicalRecord {
recordId (string): Identifies the bibliographical record for the available materials, The FAUST number,
reservations (integer): Total number of current active reservations for the bibliographical record,
reservable (boolean): True if there is any reservable materials,
holdings (array[Holdings]): An array of holdings for the materials matching the bibliographical record, as distributed across branches, departments and locations
}
Holdings {
materials (array[Material]): Materials that belongs to this placement,
location (AgencyLocation, *optional*): Placement location,
sublocation (AgencySublocation, *optional*): Placement sublocation,
department (AgencyDepartment, *optional*): Placement department,
branch (AgencyBranch): Placement branch
}
Material {
itemNumber (string): Identifies the material,
periodical (Periodical, *optional*): Present if material is a periodical,
available (boolean): True if material is available on-shelf, false if lent out,
materialGroupName (string): Name of the material group that the material belongs to
}
Periodical {
volume (string, *optional*),
volumeYear (string, *optional*),
displayText (string): A representation of the periodica volume information that is suitable for display,
volumeNumber (string, *optional*)

```
}
AgencyLocation {
  locationId (string): Location identifier,
  title (string): Name of the location
}
AgencySublocation {
  title (string): Name of the sub-location,
  sublocationId (string): Sub-location identifier
}
AgencyDepartment {
  departmentId (string): Department identifier,
  title (string): Name of the department
}
AgencyBranch {
  branchId (string): ISIL of branch (e.g. DK-761501),
  title (string): Name of branch
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

Patron (deprecated version 2)

Show/Hide | List Operations | Expand Operations | Raw

POST /external/v2/{agencyid}/patrons Create a new patron who is a person.

Implementation Notes

When a patron doesn't have a patron account in the library system, but logs in using a trusted authentication source (e.g NemId), the patron account can be created using this service. Name and address will be automatically fetched from CPR-Registry, and cannot be supplied by the client. If the CPR-Registry is not authorized to provide information about the patron, then repsonse message 404 will be sent back

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
createPatronRequest	**the patron to be created**	body	Model | Model Schema

CreatePatronRequest {
cprNumber (string),
pincode (string),
patron (PatronSettings)
}

PatronSettings {
emailAddress (string, *optional*): Required if patron should receive email notifications Existing email addresses are overwritten with this value If left empty existing email addresses are deleted,
phoneNumber (string, *optional*): Required if patron should receive SMS notifications Existing phonenumbers are overwritten with this value If left empty existing phonenumbers are deleted,
preferredPickupBranch (string): ISIL-number of preferred pickup branch,
onHold (Period, *optional*): If not set then the patron is not on hold,
receiveEmail (boolean),
receivePostalMail (boolean),
receiveSms (boolean)

Parameter	Description	Parameter Type	Data Type
			} **Period {** **from** (string, *optional*): Open-ended if not set, **to** (string, *optional*): Open-ended if not set **}**

Response Class

Model | Model Schema

AuthenticatedPatron {
authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,
patron (Patron, *optional*): Only available if patron exists in FBS and was succesfully authenticated.
}
Patron {
birthday (string, *optional*),
secondaryAddress (Address, *optional*),
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (Address, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library
}
Address {
country (string),
city (string),

```
    street (string): Street and number,
    coName (string): c/o name,
    postalCode (string)
}
Period {
    from (string, optional): Open-ended if not set,
    to (string, optional): Open-ended if not set
}
BlockStatus {
    blockedReason (string): Reason code for block,
    blockedSince (string),
    message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	Data not found	

PUT /external/v2/{agencyid}/patrons/{patronid} Update information about the patron.

Implementation Notes

The name and address cannot be supplied by the client. If the CPR-Registry is not authorized to provide information about the patron, then the name and address will not be updated.

It is possible to either update just the pincode, update just some patron settings, or update both.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion

- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
patronid	**the patron to be updated**	path	integer
updatePatron	**updated information about the patron**	body	Model | Model Schema **UpdatePatronRequest {** **patron** (PatronSettings, *optional*): Set this if patron details are to be changed, **pincodeChange** (PincodeChange, *optional*): Set this if pincode is to be changed **}** **PatronSettings {** **emailAddress** (string, *optional*): Required if patron should receive email notifications Existing email addresses are overwritten with this value If left empty existing email addresses are deleted, **phoneNumber** (string, *optional*): Required if patron should receive SMS notifications Existing phonenumbers are overwritten with this value If left empty existing phonenumbers are deleted, **preferredPickupBranch** (string): ISIL-number of preferred pickup branch, **onHold** (Period, *optional*): If not set then the patron is not on hold, **receiveEmail** (boolean), **receivePostalMail** (boolean), **receiveSms** (boolean) **}** **Period {** **from** (string, *optional*): Open-ended if not set, **to** (string, *optional*): Open-ended if not set

Parameter	Description	Parameter Type	Data Type
			} **PincodeChange {** **pincode** (string): The new pincode for the libraryCard, **libraryCardNumber** (string): Identifies the libraryCard for which the pincode is to be changed. This can be either a physical card or the CPR number, that is used as a libraryCard }

Response Class

Model | Model Schema

AuthenticatedPatron {
authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,
patron (Patron, *optional*): Only available if patron exists in FBS and was succesfully authenticated.
}
Patron {
birthday (string, *optional*),
secondaryAddress (Address, *optional*),
preferredPickupBranch (string): ISIL of preferred pickup branch,
address (Address, *optional*),
onHold (Period, *optional*): If not set then the patron is not on hold,
patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library

```
}
Address {
  country (string),
  city (string),
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string)
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	
404	patron not found	

POST /external/v2/{agencyid}/patrons/authenticate

Authenticates a patron and returns the patron details.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen

- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
authenticationRequest	**credentials for patron to be authenticated**	body	Model | Model Schema **AuthenticationRequest {** **pincode** (string): The pincode that belongs to the libraryCardNumber in plain text, **libraryCardNumber** (string): Identifies a libraryCard. This can be either a physical card or a CPR number that is used as a libraryCard **}**

Response Class

Model | Model Schema

AuthenticatedPatron {

authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,

patron (Patron, *optional*): Only available if patron exists in FBS and was succesfully authenticated.

}

Patron {

birthday (string, *optional*),

secondaryAddress (Address, *optional*),

preferredPickupBranch (string): ISIL of preferred pickup branch,

address (Address, *optional*),

onHold (Period, *optional*): If not set then the patron is not on hold,

patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,

receiveEmail (boolean),

blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library
}
Address {
country (string),
city (string),
street (string): Street and number,
coName (string): c/o name,
postalCode (string)
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
BlockStatus {
blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

POST /external/v2/{agencyid}/patrons/preauthenticated

Returns the patron details of a patron that the client has pre-authenticated using a third party.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron. Note: Whis method can only be used for patrons who are people, and not e.g. Companies or Libraries.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
cprNumber	**CPR-number of the patron**	body	string

Response Class

Model | Model Schema

AuthenticatedPatron {

authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,

patron (Patron, *optional*): Only available if patron exists in FBS and was succesfully authenticated.

}

Patron {

birthday (string, *optional*),

secondaryAddress (Address, *optional*),

preferredPickupBranch (string): ISIL of preferred pickup branch,

address (Address, *optional*),

onHold (Period, *optional*): If not set then the patron is not on hold,

patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
receiveEmail (boolean),
blockStatus (array[BlockStatus], *optional*): A list of block statuses - if the patron is not blocked then this value is empty or null,
receiveSms (boolean),
emailAddress (string, *optional*),
phoneNumber (string, *optional*),
name (string, *optional*),
receivePostalMail (boolean),
allowBookings (boolean, *optional*): True if the user is allowed to create bookings.,
defaultInterestPeriod (integer): Length of default interest period in days,
resident (boolean): True if the user is resident in the same municipality as the library
}
Address {
country (string),
city (string),
street (string): Street and number,
coName (string): c/o name,
postalCode (string)
}
Period {
from (string, *optional*): Open-ended if not set,
to (string, *optional*): Open-ended if not set
}
BlockStatus {
blockedReason (string): Reason code for block,
blockedSince (string),
message (string): Message about block
}

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	

HTTP Status Code	Reason	Response Model
401	client unauthorized	

POST /external/v2/{agencyid}/patrons/preauthenticated/unic

Returns the patron details of a patron that the client has pre-authenticated using UNIC.

Implementation Notes

The returned patron details includes a patronId that has to be used by all subsequent service calls made on behalf of that patron. Note: Whis method can only be used for patrons who are persons, and not e.g. Companies or Libraries.

If a patron is blocked the reason is available as a code:
- 'O': library card stolen
- 'U': exclusion
- 'F': extended exclusion
- 'S': blocked by self service automaton
- 'W': self created at website

The codes are informational, and can be used for looking up end user messages by the client system. However, the list is subject to change at any time, so any unexpected values should be interpreted as 'other reason'.

Parameters

Parameter	Description	Parameter Type	Data Type
agencyid	**ISIL of the agency (e.g. DK-761500)**	path	string
unicUsername	**UNIC username of the patron**	body	string

Response Class

Model | Model Schema

AuthenticatedPatron {

authenticated (boolean): True if patron successfully authenticated. If false then either the user is not known in the FBS, or an invalid combination of authentication parameters has been used.,

patron (Patron, *optional*): Only available if patron exists in FBS and was succesfully authenticated.

}

Patron {

```
  birthday (string, optional),
  secondaryAddress (Address, optional),
  preferredPickupBranch (string): ISIL of preferred pickup branch,
  address (Address, optional),
  onHold (Period, optional): If not set then the patron is not on hold,
  patronId (integer): Patron identifier to be used in subsequent service calls involving the patron,
  receiveEmail (boolean),
  blockStatus (array[BlockStatus], optional): A list of block statuses - if the patron is not blocked then this value is empty or null,
  receiveSms (boolean),
  emailAddress (string, optional),
  phoneNumber (string, optional),
  name (string, optional),
  receivePostalMail (boolean),
  allowBookings (boolean, optional): True if the user is allowed to create bookings.,
  defaultInterestPeriod (integer): Length of default interest period in days,
  resident (boolean): True if the user is resident in the same municipality as the library
}
Address {
  country (string),
  city (string),
  street (string): Street and number,
  coName (string): c/o name,
  postalCode (string)
}
Period {
  from (string, optional): Open-ended if not set,
  to (string, optional): Open-ended if not set
}
BlockStatus {
  blockedReason (string): Reason code for block,
  blockedSince (string),
  message (string): Message about block
}
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model
400	bad request	
401	client unauthorized	

[BASE URL: http://localhost:8080/externalapidocs/apidocs , API VERSION: 3]