Informat
schoolsoftware

Invoice API

Implementation guide

Web service version	1
Document version	V 4.2
Release date	June 2021
Last modified	April 2024
Last modified by	Dries Ballyn

Contents

Document description

This document describes the invoice API and provides details required for a technical implementation. It defines the communication model to use when talking to the API, including the required security algorithms. It describes all of the available methods.

Ownership

This document is written and maintained by Informat.

Change log

Date	Change	Description
14-10-2021	Credit notes	Negative values in qty, unit & total of items are returned
14-10-2021	Archived address	Details of archived invoice addresses are returned
14-10-2021	Create PDF	Creating PDFs is now possible via the API
21-02-2022	Nuget update 12.6.0	500 status code for all graphQL calls due to error “Parameter count mismatch”. Fix https://github.com/ChilliCream/hotchocolate/pull/4182
25-08-2022	Payment support	New API endpoint to register manual payments for invoices/credit notes in Informat (incl. BadDebt indicator)
25-08-2022	VZW validation	VZW property of scope/access group is verified. From now on, the API returns only invoices that have been booked into the VZW that is set in the scope/access group.
25-08-2022	Get invoice	Support to get the data of one invoice without the need to specify invoice group Id
25-08-2022	discountAmount	Change calculation of discountAmount (price * qty * percentage) to prevent that rounding to 2 decimals results in a discountAmount when there is no discountPercentage
10-01-2023	Create PDF	Added support to generate PDF of credit notes (Rekboek Transaction Worker update)
03-04-2023	Payment registration	Bugfix related to date format of user data (Last Logon) when trying to register payments
01-06-2024	Mandates	Return mandate details for an invoice/credit note
01-06-2024	Relations	Return LPV1 & LPV2 relations associated with the address(es)

Basics

Endpoint

The **production environment** is available via this endpoint:

`https://invoiceapi.informatsoftware.be`

How to obtain credentials?

You need to submit a request to the Informat helpdesk to obtain credentials for your client:

- Informat will provide a unique ClientId & ClientSecret to authenticate your application with the Invoice API
- You also need scopes to read/write data on behalf of iRekeningen access group(s) of the school.

See How to get access to invoice data? on page 9

Character sets

All service response objects will be formatted with UTF-8 encoding to ensure compatibility with most languages.

Request Format and Responses

Request verbs

All request data should be specified in the JSON format, except when stated otherwise.

JSON Basics

The majority of requests and responses to the WEB API use the JavaScript Object Notation (JSON) for formatting the content and structure of the data and responses.

JSON is used because it is the simplest and easiest solution for working with data within a web browser, as JSON structures can be evaluated and used as JavaScript objects within the web browser environment.

JSON supports the same basic types as supported by JavaScript, these are:

- **Number** (either integer or floating-point).
- **String**; this should be enclosed by double-quotes and supports Unicode characters and backslash escaping. For example: `"A String"`
- **Boolean** - a true or false value. You can use these strings directly. For example: `{ "value": true }`
- **Array** - a list of values enclosed in square brackets. For example: `["one", "two", "three"]`
- **Object** - a set of key/value pairs (i.e. an associative array, or hash). The key must be a string, but the value can be any of the supported JSON values. For example:

```
{
   "servings" : 4,
   "subtitle" : "Easy to make in advance, and then cook when ready",
   "cooktime" : 60,
   "title" : "Chicken Coriander"
}
```

Handling dates (and time)

Dates should always be sent to the server in either UTC format or using the yyyy/mm/dd (hh:mm:ss) notation.

Example:

```
2019-09-30T15:30:22.000Z            // Valid UTC Format
2019-09-30T15:30:22+0100            // Valid UTC Format
2019/09/30
2019/09/30 15:30:22
```

If you're using javascript Date() objects in your request body, your browser will automatically convert these to the UTZ format.

Example:

```
// javascript
var someDate = new Date(2019,05,21)    // will be sent as 2019-05-21T00:00:00.000Z
```

If you need to send a date and time object, use the UTC format or use yyyy/mm/dd hh:mm:ss

Example:

```
2019-09-30T22:30:00.000Z            // UTC Format
2019/09/30 22:30:00
```

Boolean

Booleans should always be formatted as true/false (lower case, no quotation marks).

Example:

```
{
   "isAdmin": true,
   "canAccess": false
}
```

Response format

The API can only return data in JSON format, except when stated otherwise.

HTTP Headers

Because the API uses HTTP for all communication, you need to ensure that the correct HTTP headers are supplied (and processed on retrieval) so that you get the right format and encoding. Different environments and clients will be more or less strict on the effect of these HTTP headers (especially when not present). Where possible you should be as specific as possible.

Request Headers

`Content-type`

Specifies the content type of the information being supplied within the request. The specification uses MIME type specifications. For the majority of requests this will be JSON (`application/json`).

The use of the correct `Content-type` header on a request is required, unless the body is empty.

`Accept`

Specifies the list of accepted data types to be returned by the server (i.e. that are accepted/understandable by the client). The format should be a list of one or more MIME types, separated by colons.

For the majority of requests the definition should be for JSON data (`application/json`).

The use of `Accept` in queries is not required, but is highly recommended as it helps to ensure that the data returned can be processed by the client.

Response Headers

Response headers are returned by the server when sending back content and include a number of different header fields, many of which are standard HTTP response header and have no significance to operation. The list of response headers are listed below.

HTTP Status Codes

With the interface to invoices working through HTTP, error codes and statuses are reported using a combination of the HTTP status code number, and corresponding data in the body of the response data.

A list of the error codes returned by the API, and generic descriptions of the related errors are provided below. The meaning of status codes for specific request types is provided in the corresponding API call reference.

Code	Text	Description
200	OK	Request completed successfully. The body contains any response data.
400	Bad Request	Bad request structure. The error can indicate an error with the request URL, path or headers. Differences in the supplied MD5 hash and content also trigger this error, as this may indicate message corruption.
401	Unauthorized	The item requested was not available using the supplied authorization, or authorization was not supplied.
403	Forbidden	The requested item or operation is forbidden. E.g. - InvoiceId is out of scope / no access allowed
404	Not Found	The requested content could not be found. E.g. - InvoiceId is in scope but not booked
405	Resource Not Allowed	A request was made using an invalid HTTP request type for the URL requested. For example, you have requested a PUT when a POST is required. Errors of this type can also triggered by invalid URL strings.
406	Not Acceptable	The requested content type is not supported by the server.
409	Conflict	Request resulted in an update conflict.
415	Bad Content Type	The content types supported, and the content type of the information being requested or submitted indicate that the content type is not supported.
416	Requested Range Not Satisfiable	The range specified in the request header cannot be satisfied by the server.
500	Internal Server Error	The request was invalid, either because the supplied JSON was invalid, or invalid information was supplied as part of the request.

Usage restrictions

To be able to balance the load on our servers and the impact on our overall performance, this service requires client implementations to comply with some basic rules. In case a specific resource or method requires additional rules, these will be added to the corresponding chapters.

Ground rules

- Cache retrieved data as much as possible, use the provided ID's and references to maintain data consistency.
- The privacy and security of the private key is the responsibility of the end user. In case the private key is compromised, the end user is required to inform Informat as soon as possible. We will then take the necessary steps to ensure the safety of the data. A new private key will then be issued.

Privacy concerns

The user is responsible for the safe and correct processing of all personal data, conform privacy regulations, as is stipulated in the contract between the user and the supplier, Informat.

Disclaimer

All requests to the service are logged and monitored to aid in the improvement of the service, to help troubleshoot issues and to avoid abuse.

If Informat finds that the user is found guilty of abuse, Informat reserves the right to terminate the users' access to the service if the abuse continues after multiple warnings.

Security

Transport security

Secure Socket Layer

To ensure the safety of the transferred data and to prevent data-theft, this service operates only via HTTPS.

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols designed to provide communication security over the Internet. They use X.509 certificates and hence asymmetric cryptography to authenticate the counterparty with whom they are communicating, and to exchange a symmetric key. This session key is then used to encrypt data flowing between the parties. This allows for data/message confidentiality, and message authentication codes for message integrity and as a by-product, message authentication.

Authentication endpoint

The **production environment** is available via:

`identityEndpoint = https://www.identityserver.be/connect/token`

Remark: *On acceptance environment, use https://acc.identityserver.be/connect/token*
Mind that client Id and Secret for ACC and Production are different.

Request authentication

Modern secure applications often use access tokens to ensure a user has access to the appropriate resources, and these access tokens typically have a limited lifetime. This is done for various security reasons: for one, limiting the lifetime of the access token limits the amount of time an attacker can use a stolen token. Also, the information contained in or referenced by the access token could become stale.

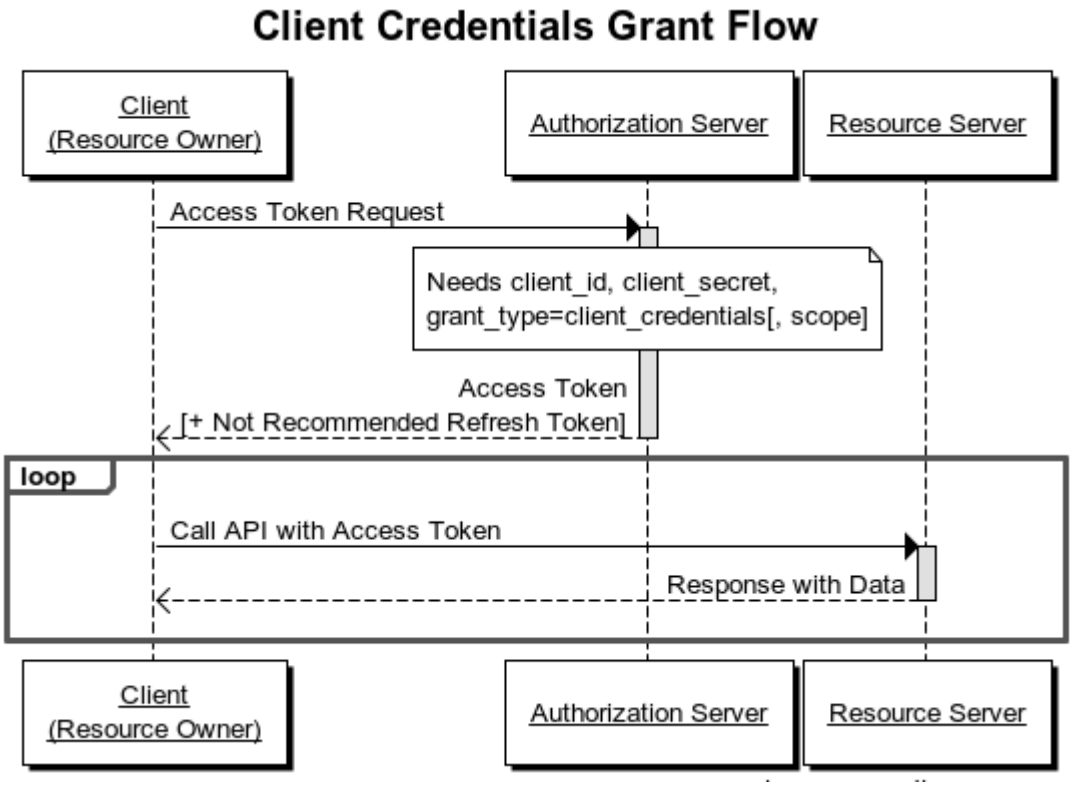
Calling resources without Access Token

When trying to access the API without Access Token, an Unauthorized response will be returned.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer
```

Getting an Access Token by client credentials

The Client Credentials grant type is used by clients to obtain an access token by using a clientId and clientSecret.



How to get access to invoice data?

You need a scope to read/write data using the Invoice API.
A scope is a permission that is set on a token, a context in which that token may act.

Scopes are created at the level of iRekeningen access groups (= toegangsgroepen). An access group grants users access to the invoice data of the connected schools, school locations, boarding schools or institutions.

1) Ask the iRekeningen administrator of the school to identify (or create) the access group(s) covering the schools of your interest, i.e. to which API access is requested.

   ***Attention**: Mind that the VZW property of the access group needs to match the administration into which the invoices were booked to be able to retrieve them using the Invoice API.*

2) Provide the names of the iRekeningen access groups to the Informat helpdesk.
3) Informat will provide a scope for each access group.

For example, a token with the **api_informat_invoices.invoices.read.2949957b-5dc1-4acc-a7cf-ba322b62cb12_123456** scope is permitted to read data from access group 2949957b-5dc1-4acc-a7cf-ba322b62cb12 in school database with installation ID 123456. Otherwise you would be denied access.

Request token for 1 scope

Request

```
POST https://www.identityserver.be/connect/token HTTP/1.1
...
Content-Type: application/x-www-form-urlencoded

grant_type: client_credentials
client_id: <received client id>
client_secret: <received client secret>
scope: api_informat_invoices.invoices.read.2949957b-5dc1-4acc-a7cf-ba322b62cb12_123456
```

Response

```
HTTP/1.1 200 OK
...

{
"access_token":"……",
"expires_in": 3600,
"token_type": "Bearer",
"scope": "api_informat_invoices.invoices.read.2949957b-5dc1-4acc-a7cf-ba322b62cb12_123456"
}
```

Request token for multiple scopes

Doing a request for multiple scopes is currently not possible.

Use of access token

Each API Request must be sent with the access token as follows:

Authorization: BEARER <token>

Example

```
GET https://invoiceapi.informatsoftware.be/... HTTP/1.1
Authorization: BEARER <token from identityserver>
```

Resources

This chapter lists all the resources available on the webservice. A resource is an entity which you can get data for.

GraphQL

This chapter describes how to query the api via the graphQL endpoint. It describes all the available methods, their request/response format and provides input/output examples.

Informat has chosen the GraphQL technology to expose the invoice data to third parties. See https://graphql.org/ for more information about this technology.

With graphQl you define a query in the body of the request. There is a “maximum” data shape defined, with a “maximum” set of filters. You can define which fields you want to have returned by the endpoint yourself, with the filters you want, all from the maximum set that is available.

The root object of the query has a required parameter `accessGroupIdentifier` which needs to correspond with the accessGroupIdentifier scope you used to get the access token.

There is code completion to construct the query, which you can instantly execute. This url will later on be referred to as the graphQL playground.

An overview of the data is available here:

https://invoiceapi.informatsoftware.be/graphql

https://invoiceapi.informatsoftware.be/ui/voyager

Data & actions

For more information about the available datasets, see:

- “Invoice group dataset” on page 13
- “Invoice dataset” on page 23

Other actions that can be executed using the API interface:

- Add payment for an invoice, see page 26
- Generate PDF of invoice, see page 30
- Get PDF of invoice, see page 29

Get invoice groups

You can retrieve the list of invoice groups (with invoices) in a school year.

Mind that school year format is “2020-21” for invoice groups of type Student, Staff and Intern.
For type Registration, school year format shall be “01/09/2020-31/08/2021”.

Sample query (full dataset)

This query returns the invoice groups of type Student, Staff and Intern in the specified school year 2020-21:

GraphQL variables:

```
{
  "schoolYear": "2020-21",
  "accessGroupIdentifier": "one_of_my_access group_identifiers"
}
```

***Remark**: To retrieve invoice groups of type Registration, the school year should be formatted as “01/09/2020-31/08/2021”.*

GraphQL query:

```
query accessGroup($schoolYear: String!, $accessGroupIdentifier: String!) {
  accessGroup(accessGroupIdentifier: $accessGroupIdentifier) {
    guid
    name
    invoiceGroups(schoolYear: $schoolYear) {
      id
      name
      fromDate
      toDate
      invoiceGroupType
      schoolYear
    }
  }
}
```

Sample query for specific type

This query returns the invoice groups of type Student in the specified school year:

```
query accessGroup($schoolYear: String!, $accessGroupIdentifier: String!) {
  accessGroup(accessGroupIdentifier: $accessGroupIdentifier) {
    invoiceGroups(schoolYear: $schoolYear where: { invoiceGroupType: {eq: STUDENT}}) {
      id
      name
      invoiceGroupType
    }
  }
}
```

Sample response

This is a sample response for getting invoice groups in a school year:


```
{
    "data": {
        "accessGroup": {
            "guid": "f451ca37-411c-49ea-a544-29598ea22b82",
            "name": "ToegangsgroepABC",
            "invoiceGroups": [
                {
                    "id": 8052,
                    "name": "Rekening SEPTEMBER 2020",
                    "fromDate": "2020-09-01",
                    "toDate": "2020-09-30",
                    "invoiceGroupType": "STUDENT",
                    "schoolYear": "2020-21"
                },
                {
                    "id": 8053,
                    "name": " Rekening OKTOBER 2020",
                    "fromDate": "2020-10-01",
                    "toDate": "2020-10-25",
                    "invoiceGroupType": "STUDENT",
                    "schoolYear": "2020-21"
                },
                …
```


Invoice group dataset

The following data can be returned for each invoice group:

Field	Entity	Description
Id	invoiceGroup	Unique identifier of the invoice group in the database (p_infrek). This Id is required to retrieve the booked invoices/credit notes.
Name	invoiceGroup	Title of the invoice group
fromDate	invoiceGroup	Start date of the invoice group's scope
toDate	invoiceGroup	End date of the invoice group's scope
invoiceGroupType	invoiceGroup	STUDENT, STAFF, INTERN, REGISTRATION
schoolYear	invoiceGroup	School year as "2023-24". Use "01/09/2023-31/08/2024" if type = REGISTRATION

Get all invoices of an invoice group

You can retrieve a list of all booked invoices and credit notes in an invoice group (invoiceGroupId).
The list can be filtered according to graphQl guidelines. See page 21 for more information about the available data.

Sample query

This query returns details of all booked invoices & credit notes from the specified invoiceGroupId.

***Note**: A non-existing / out-of-scope invoiceGroupId will result in status 200 OK with an empty body.*

GraphQl variables:

```
{
  "schoolYear": "2020-21",
  "accessGroupIdentifier": "one_of_my_access group_identifiers",
  "invoiceGroupId": 1234
}
```

GraphQl query:

```
query accessGroup($schoolYear: String!, $accessGroupIdentifier: String!, $invoiceGroupId: Int!){
accessGroup(accessGroupIdentifier: $accessGroupIdentifier) {
    guid
    name
    invoiceGroups(schoolYear: $schoolYear where:{id: {eq: $invoiceGroupId}}) {
      id
      name
      fromDate
      toDate
      invoiceGroupType
      schoolyear

      invoices {
        invoiceId
        invoiceType
        printDate
        dueDate
        total
        paid
        open
        paymentReference
        paymentReferenceInvoice
        eolEntryNumber
        created
        changed
        imageUrl

        recipient {
          id
          databaseId
          firstName
          lastName
          birthDate
          stamNumber
          location
          institute
          class
          classNumber
          code
          rrNo
          bisNo
        }
```

```
        mandate {
          beginDate
          creditorId
          ibanSchool
          mandateReference
          ibanParent
          bicParent
        }

        invoiceAddress {
          databaseId
          addressGuid
          title
          name
          street
          number
          busNumber
          zip
          city
          emailAddresses
          relations {
            name
            title
            nationalNumber
            lpv
          }
        }

        homeAddress {
          databaseId
          addressGuid
          title
          name
          street
          number
          busNumber
          zip
          city
          emailAddresses
          relations {
            name
            title
            nationalNumber
            lpv
          }
        }

        itemGroups {
          code
          description
          glAccountCode
          codeBudgetExtern
          total
          items {
            description
            extraName
            quantity
            unitPrice
            discountPercentage
            discountAmount
            total
          }
        }
      }
    }
  }
}
```

Sample query with simple filter

This query returns all booked invoices & credit notes in an invoice group with open amount different from zero (= not fully paid):

```
query accessGroup($schoolYear: String!, $accessGroupIdentifier: String!, $invoiceGroupId: Int!){
accessGroup(accessGroupIdentifier: $accessGroupIdentifier) {
    invoiceGroups(schoolYear: $schoolYear where:{id: {eq: $invoiceGroupId}}) {
      invoices (where: {open: {neq: 0}}) {
        invoiceId
        invoiceType
        printDate
        dueDate
        total
        paid
        open
        paymentReference
        paymentReferenceInvoice
        eolEntryNumber
        created
        changed
      }
    }
  }
}
```

This query returns all booked invoices & credit notes of an invoice group with a changed date after June 1st, 2023: typically for booked invoices the changed date is updated when a payment was registered in Informat.

```
query accessGroup($schoolYear: String!, $accessGroupIdentifier: String!, $invoiceGroupId: Int!){
accessGroup(accessGroupIdentifier: $accessGroupIdentifier) {
    invoiceGroups(schoolYear: $schoolYear where:{id: {eq: $invoiceGroupId}}) {
      invoices (where: {changed: {gte: "2023-06-01"}}) {
        invoiceId
        invoiceType
        printDate
        dueDate
        total
        paid
        open
        paymentReference
        paymentReferenceInvoice
        eolEntryNumber
        created
        changed
      }
    }
  }
}
```

Sample query with complex filter

This query returns all booked invoices & credit notes in the specified invoice group with open amount not zero and total amount greater than or equal to 25,5 €:

```
query accessGroup($schoolYear: String!, $accessGroupIdentifier: String!, $invoiceGroupId: Int!){
accessGroup(accessGroupIdentifier: $accessGroupIdentifier) {
    invoiceGroups(schoolYear: $schoolYear where:{id: {eq: $invoiceGroupId}}) {
      invoices (where: {and: [ {open: {neq: 0}}, {total: {gte: 25.5}}]}) {
        invoiceId
        invoiceType
        printDate
        dueDate
        total
        paid
        open
        paymentReference
        paymentReferenceInvoice
        eolEntryNumber
        created
        changed
      }
    }
  }
}
```

Sample query with filter combination “and” & “or”

This query will return all booked invoices that were changed after May 1st 2020, OR that have an open amount that is not zero and a total that is greater than or equal to 25,5€:

```
query accessGroup($schoolYear: String!, $accessGroupIdentifier: String!, $invoiceGroupId: Int!) {
  accessGroup(accessGroupIdentifier: $accessGroupIdentifier) {
    invoiceGroups(schoolYear: $schoolYear, where: {id: {eq: $invoiceGroupId}}) {
      invoices(where: {or: [
        {and: [
          {open: {neq: 0}},
          {total: {gte: 25.5}}
          ]
        },
        {changed: {gte: "2020-05-01"}}
        ]}) {
        invoiceId
        …
      }
    }
  }
}
```

Get invoice

You can request one booked invoice or credit note without specifying invoice group. The response can include these data:

- Invoice data (amount, OGM, due date …)
- URL to the individual PDF of the original invoice
- Recipient details (person data)
- Mandate details (if person has mandate for bank account number associated with the layout of the booked invoice)
- Invoice address (incl. any associated email addresses & LPV relations)
- Home address (incl. any associated email addresses & LPV relations)
- Invoice lines: invoiced item groups & article details

Sample query (full dataset)

This query returns the full dataset of one invoice (without specifying the InvoiceGroupId):

GraphQl variables:

```
{
  "accessGroupIdentifier": "one_of_my_access group_identifiers",
  "invoiceId": "guid of the invoice"
}
```

GraphQl query:

```
query accessGroup($accessGroupIdentifier: String!, $invoiceId: UUID!) {
  accessGroup(accessGroupIdentifier: $accessGroupIdentifier) {
    invoice(invoiceId: $invoiceId) {
      invoiceGroup {
        id
        name
        invoiceGroupType
        schoolYear
      }
      invoiceId
      invoiceType
      printDate
      dueDate
      total
      paid
      open
      paymentReference
      paymentReferenceInvoice
      eolEntryNumber
      created
      changed
      createImageUrl
      imageUrl

      recipient {
          id
          databaseId
          firstName
          lastName
          birthDate
          stamNumber
          location
          institute
          class
          classNumber
          code
          rrNo
          bisNo
        }
```

```
        mandate {
          beginDate
          creditorId
          ibanSchool
          mandateReference
          ibanParent
          bicParent
        }

        invoiceAddress {
          databaseId
          addressGuid
          title
          name
          street
          number
          busNumber
          zip
          city
          emailAddresses
          relations {
            name
            title
            nationalNumber
            lpv
          }
        }

        homeAddress {
          databaseId
          addressGuid
          title
          name
          street
          number
          busNumber
          zip
          city
          emailAddresses
          relations {
            name
            title
            nationalNumber
            lpv
          }
        }

        itemGroups {
          code
          description
          glAccountCode
          codeBudgetExtern
          total
          items {
            description
            extraName
            quantity
            unitPrice
            discountPercentage
            discountAmount
            total
          }
        }
    }
  }
}
```

Sample response

This is a sample response for 1 invoice from an invoice group of type STUDENT:


```
{
                        "invoiceId": "cac965e9-e889-4dd5-a3d7-a7a25dfb31f8",
                        "invoiceType": "INVOICE",
                        "printDate": "2020-10-31",
                        "dueDate": "2020-11-30",
                        "total": 65.10,
                        "paid": 0.00,
                        "open": 65.10,
                        "paymentReference": "000282332442",
                        "paymentReferenceInvoice": null,
                        "eolEntryNumber": null,
                        "created": "2020-10-13T12:02:03.067Z",
                        "changed": "2020-11-16T10:38:31.797Z",
                        "imageUrl": null,
                        "recipient": {
                            "id": "9f7cf737-bbd3-4333-84ea-f145cc722591",
                            "databaseId": 67689,
                            "firstName": "Mats",
                            "lastName": "Verheghe",
                            "birthDate": "2010-02-03",
                            "stamNumber": "201200008",
                            "institute": "016981",
                            "location": "post 12",
                            "class": "L5C",
                            "classNumber": 2,
                            "code": null,
                            "rrNo": "10020342232",
                            "bisNo": null
                        },
                        "mandate": {
                            "beginDate": "2021-04-16",
                            "creditorId": "BE78ZZZ0447943822",
                            "ibanSchool": "BE94.0622.1062.4514",
                            "mandateReference": "2",
                            "ibanParent": "BE13.6685.9758.8700",
                            "bicParent": "GKCCBEBB"
                        },
                        "invoiceAddress": {
                            "databaseId": 75215,
                            "addressGuid": "5e2cde0d-bfd9-41ee-8f18-6c6e27a672c2",
                            "title": "Aan de vader van",
                            "name": " Mats Verheghe",
                            "street": "Vaartstraat",
                            "number": "8",
                            "busNumber": "",
                            "zip": "8600",
                            "city": "DIKSMUIDE",
                            "emailAddresses": “jasper.verheghe@gmail.com”
                            "relations": [
                                {
                                    "name": "Verheghe Harald",
                                    "title": "Vader",
                                    "nationalNumber": null,
                                    "lpv": 2
                                },
                                {
                                    "name": "Wolkarius Ann",
                                    "title": "Moeder",
                                    "nationalNumber": "84080651242",
                                    "lpv": 1
                                }
                            ]
                        },
```

```
                "homeAddress": {
                    "databaseId": 170465,
                    "addressGuid": "9fa516bd-f5ff-431a-87b5-ef18f50d04e0",
                    "title": "Aan de ouder(s) van",
                    "name": "Mats Verheghe",
                    "street": "Wallestraat",
                    "number": "15",
                    "busNumber": "A",
                    "zip": "8820",
                    "city": "TORHOUT",
                    "emailAddresses": “jasper.verheghe@gmail.com,ann@homedecor.be”
                    "relations": [
                        {
                            "name": "Verheghe Harald",
                            "title": "Vader",
                            "nationalNumber": null,
                            "lpv": 2
                        },
                        {
                            "name": "Wolkarius Ann",
                            "title": "Moeder",
                            "nationalNumber": "84080651242",
                            "lpv": 1
                        }
                    ]
                },
                "itemGroups": [
                    {
                        "code": "MEEUIT",
                        "description": "Meerdaagse uitstappen",
                        "glAccountCode": "700700",
                        "codeBudgetExtern": "2021/700700",
                        "total": 26.00,
                        "items": [
                            {
                                "description": "Openluchtklassen",
                                "extraName": "1e schijf",
                                "quantity": 1.00,
                                "unitPrice": 26.00,
                                "discountPercentage": 0.00,
                                "discountAmount": 0.00,
                                "total": 26.00
                            }
                        ]
                    },
                    {
                        "code": "MTZ",
                        "description": "Middagtoezicht",
                        "glAccountCode": "703460",
                        "codeBudgetExtern": "2021/703460",
                        "total": 14.40,
                        "items": [
                            {
                                "description": "Middagtoezicht",
                                "extraName": "",
                                "quantity": 9.00,
                                "unitPrice": 1.60,
                                "discountPercentage": 0.00,
                                "discountAmount": 0.00,
                                "total": 14.40
                            }
                        ]
                    },
                    {
                        "code": "NASOPV",
                        "description": "Naschoolse opvang",
                        "glAccountCode": "703470",
                        "codeBudgetExtern": "2021/703470",
                        "total": 3.20,
```

```
                "items": [
                    {
                        "description": "Naschoolse opvang",
                        "extraName": "",
                        "quantity": 14.00,
                        "unitPrice": 0.20,
                        "discountPercentage": 0.00,
                        "discountAmount": 0.00,
                        "total": 2.80
                    },
                    {
                        "description": "Studie",
                        "extraName": "",
                        "quantity": 2.00,
                        "unitPrice": 0.20,
                        "discountPercentage": 0.00,
                        "discountAmount": 0.00,
                        "total": 0.40
                    }
                ]
            }
        ]
    },
…
```


Invoice dataset

The following data can be returned for each invoice/credit note:

Field	Entity	Description
invoiceId	Invoice	Unique identifier of the invoice or credit note across databases
invoiceType	Invoice	INVOICE or CREDIT_NOTE
printDate	Invoice	Print date (reference to find valid student subscription to determine stamNumber, class, class number)
dueDate	Invoice	Due date of the invoice. Identical to print date for credit notes
total	Invoice	Total amount (credit notes = positive amount)
paid	Invoice	Amount that's already paid
open	Invoice	Amount still due
paymentReference	Invoice	12-digit payment reference (only digits)
paymentReferenceInvoice	Invoice	Only for credit notes: 12-digit payment reference of related invoice
eolEntryNumber	Invoice	Exact sales entry id of invoice transferred to Exact Online
created	Invoice	Date/time of invoice creation
changed	Invoice	Date/time of last change
createImageUrl	Invoice	If ImageUrl is NULL (= PDF does not exist), use this URL to send create PDF command. See Pdf POST method
imageUrl	Invoice	Url to get PDF of the invoice from Azure blob storage, if it exists. See Pdf GET method on page 19.
id	Recipient	Unique identifier of the recipient across databases (rowguid)
databaseId	Recipient	Identifier of the recipient in this database (p_persoon)
firstName	Recipient	First name of the recipient
lastName	Recipient	Family name of the recipient

Field	Entity	Description
birthdate	Recipient	Birth date of the recipient
stamNumber	Recipient	Official student ID (issued by Dpt. of Education). Based on student subscription at print date. (only if Invoice group type = STUDENT)
location	Recipient	Code identifying the official location (“vestcode”) in which the invoice was generated. Based on layout attached to invoice (only if Invoice group type = STUDENT or INTERN)
institute	Recipient	Official institute number of the (boarding) school in which the invoice was generated. For invoice group type REGISTRATION, this is the name of the institution. Based on layout attached to invoice.
class	Recipient	Class in which student has subscription at print date (only if Invoice group type = STUDENT)
classNumber	Recipient	Class number (only if Invoice group type = STUDENT)
code	Recipient	8-digit “Code Analyt. Boekh.” of the official class, if any (only if Invoice group type = STUDENT)
rrNo	Recipient	Official social security number if Belgian student
bisNo	Recipient	Official number if non-Belgian student
beginDate	Mandate	Begin date of valid mandate associated with the bank account number of the school of the layout stored with the booked invoice
creditorId	Mandate	Creditor identification number of the school
ibanSchool	Mandate	Bank account number of the school (layout settings)
mandateReference	Mandate	Mandate reference (without BEMDT prefix, if applicable)
ibanParent	Mandate	Bank account number of the parent
bicParent	Mandate	BIC code of the parent’s bank
databaseId	invoiceAddress	Identifier of the invoice address in this database (p_adres). This is the address at the time the invoice was created. If deleted at the time of the API request, historical address data shall be returned.
addressGuid	invoiceAddress	Unique identifier of the address across databases (adressenuuid)
title	invoiceAddress	Title assigned to the name of the address in MIS/SAS, e.g. Aan de ouder(s) van, Aan de vader van, Aan de moeder van, etc
name	invoiceAddress	Name assigned to the address in MIS/SAS. This can be student’s name (in combination with Title “Aan de ouders van”) or the name of one of the parents
street	invoiceAddress	Street name
number	invoiceAddress	Street number
busNumber	invoiceAddress	Bus number, if any
zip	invoiceAddress	Zip
city	invoiceAddress	City name
emailAddresses	invoiceAddress	Comma-separated list of email addresses active for this address
name	Relation	Name of LPV relation associated with the invoice address in MIS/SAS (can be max. 2)
title	Relation	Role of relation (father, mother, ...)
nationalNumber	Relation	rrNo of relation/parent

Field	Entity	Description
lpv	Relation	1 or 2 (LPV = leerplichtverantwoordelijke)
databaseId	homeAddress	Identifier of the home address in this database (p_adres). This is the home address of the person at the time of the API request. Can be identical to InvoiceAddress.
addressGuid	homeAddress	Unique identifier of the address across databases
title	homeAddress	Title assigned to the name of the address in MIS/SAS
name	homeAddress	Name assigned to the address in MIS/SAS
street	homeAddress	Street name
number	homeAddress	Street number
busNumber	homeAddress	Bus number, if any
zip	homeAddress	Zip
city	homeAddress	City name
emailAddresses	homeAddress	Comma-separated list of email addresses active for this address
name	Relation	Name of LPV relation associated with the home address in MIS/SAS (can be max. 2)
title	Relation	Role of relation (father, mother, ...)
nationalNumber	Relation	rrNo of relation/parent
lpv	Relation	1 or 2 (LPV = leerplichtverantwoordelijke)
itemGroups		Collection of item groups (= hoofdrubrieken) on this invoice
code	ItemGroup	Code of the item group
description	ItemGroup	Description of the item group
glAccountCode	ItemGroup	G/L account assigned to the item group at the time of retrieval (not the one in document in Informat created at booking)
codeBudgetExtern	ItemGroup	External reference (glAccountCode + BBC export parameters in layout)
total	ItemGroup	Total cost of items in this group (can be negative!)
items	ItemGroup	Collection of items in this item group
description	ItemGroupItem	Name of the invoiced item
extraName	ItemGroupItem	Extra description added to item name, if any
quantity	ItemGroupItem	Quantity (can be negative!)
unitPrice	ItemGroupItem	Price (can be negative!)
discountPercentage	ItemGroupItem	Discount percentage. 0.00 if no discount
discountAmount	ItemGroupItem	Discount amount. 0.00 if no discount
total	ItemGroupItem	Total cost of this item (can be negative!)

Add payment endpoint

You can register manual payments for invoices/credit notes.

There is a swagger endpoint that gives you more information about the usage of this endpoint

`https://invoiceapi.informatsoftware.be/swagger/index.html`

Prerequisites

The following prerequisites apply when registering payments using the API:

- A journal named “Extern” (and “Extern_Dubieus”) should exist in the Informat “deelboek” where the invoices/credit notes are booked. To be requested via Informat helpdesk.
- An active user account shall be assigned to the access group / scope that is used. To be added by school administrator.

Request

Headers

Besides the mandatory headers, no other custom headers are required.

URL syntax

https://invoiceapi.informatsoftware.be/api/payments/{accessGroupIdentifier}/{invoiceId}

- accessGroupIdentifier: the access group identifier used for the access token
- invoiceId: the id (guid) of the invoice or credit note

Body

The body contains the required payment data:

```
{"invoiceId":"9a783289-b3e2-44d9-8ff5-cc7d5460dcae","paymentReference":"000198647108",
"amountPaid":10.50, "datePaid":"2022-06-09", "isBadDebt":false}
```

- invoiceId: the id (guid) of the invoice or credit note (see also URL)
- paymentReference: the 12-digit OGM of the invoice or credit note
- amountPaid: pos/neg amount of the payment transaction. Always use “period” as decimal separator, e.g. 10.50:
    - Positive amount: invoice = (partial) payment / credit note = (partial) reopen
    - Negative amount: invoice = (partial) reopen / credit note = (partial) refund
- datePaid: date of payment
- isBadDebt: true or false (optional parameter – used for registering “lost” payments in separate journal)

Sample

URL:

…/api/payments/943d5e11-9910-4251-bfce-d1e64c533cc3_502052/9a783289-b3e2-44d9-8ff5-cc7d5460dcae

Body:

```
{"invoiceId":"9a783289-b3e2-44d9-8ff5-cc7d5460dcae", "paymentReference":"000198647108",
"amountPaid":10.0, "datePaid":"2022-06-09"}
```

Response

Success

The response will be a status code 200 (OK) if payment was registered successfully.
The body contains the following data:

```
{
"totalAmount": 12.00,
"totalPaid": 5.00,
"openAmount": 7.00,
"invoiceId":"9a783289-b3e2-44d9-8ff5-cc7d5460dcae",
"paymentReference":"000198647108"
}
```

- totalAmount: amount of the invoice or credit note
- totalPaid: sum of all registered payments for this invoice
- openAmount: open unpaid amount or balance if negative
- invoiceId: the id (guid) of the invoice or credit note
- paymentReference: the 12-digit OGM of the invoice or credit note

Failure

The following error codes can be returned if payment registration fails:

Code	Text	Description
400	Bad Request	Bad request structure. The error can indicate an error with the request URL, path or headers. Error message in body can be as follows: - Invoice – OGM mismatch (= OGM not correct for invoiceId) - Invoice not found (= not booked yet) - Period not found for <datePaid> in deelboek <deelboek> - Journal Extern / Extern_Dubieus not found in deelboek <deelboek> - Section Id not found (multiple journals Extern in VZW)
403	Forbidden	The requested item or operation is forbidden. - InvoiceId is out of scope / no access allowed
500	Internal Server Error	The request was invalid, either because the supplied JSON was invalid, or invalid information was supplied as part of the request.

How it works

Exact Online *Mind that there is no use in registering payments via the API if the Exact Online integration is active for a school. The Exact synchronization in Rekeningen 2.0 will overrule any changes that were made via the API.*

Amount

The amount passed in the method determines the payment transaction that is registered for the corresponding invoice.

Attention*: Avoid duplicate payment requests for the same invoice because payment will be registered multiple times. If invoice is unpaid, don't send payment request because payment will be registered for this invoice for paidAmount.*

E.g. if invoice has open amount of 15 €, and amountPaid of 15 € is passed, the invoice will have a remaining open amount of 0 €. If credit note has open amount of -10 €, and amountPaid of -10 € (= refund) is passed, the credit note will have a remaining open amount of 0 €.

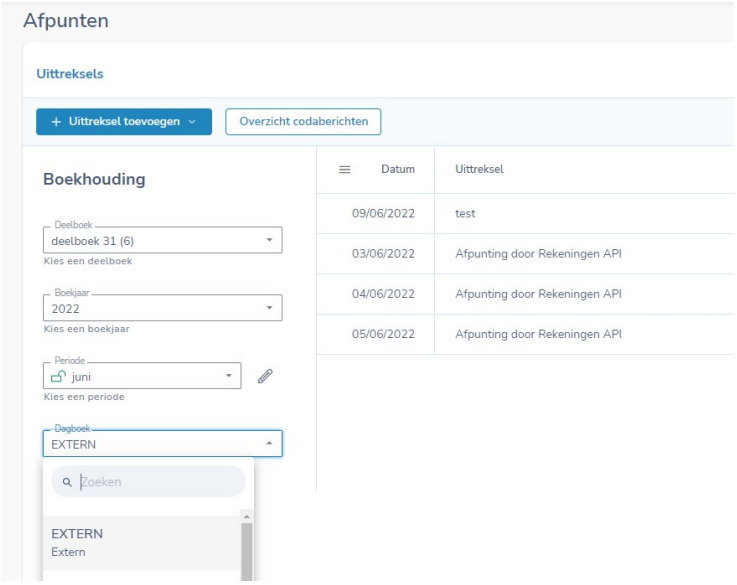
This table provides overview of different results depending on initial open amount & transaction amount :

Initial open amount	amountPaid passed in method	Remaining open amount
15	15	0
15	12	3
3	-12	15
15	20	-5
-5	5	-10
-10	-10	0
15	15 15	-15

Track & trace in Rekeningen 2.0

The Invoice API will insert a manual payment in the period that corresponds to the datePaid in the request.
The payments are always inserted into journal "Extern" of the "deelboek" where the invoice or credit note was booked if parameter "IsBadDebt" is False or omitted from the request. When IsBadDebt is True, the payment is inserted into journal "Extern_Dubieus" which allows to differentiate between the actual received payments and invoices that are considered "lost" (see reports "Boekingen per persoon" & "Journaal" in Rekeningen 2.0).

The API payments can be checked via the *Afpunten* page in the Rekeningen application:



When no statement is found for the specified datePaid in the request, it will be added automatically with the following name "Afpunting door Rekeningen API".

Get/Create PDF endpoint

For every invoice, an imageUrl can be returned. It refers to the original PDF of the individual invoice or credit note on Azure.

There is an endpoint where you can

- Get the PDF (if it exists)
- Send a create command that asynchronously tries to create the PDF

The endpoints need an access token for a specific access group identifier, the same way the graphql endpoint needs one.

> There is a swagger endpoint that gives you more information about the usage of this endpoint
>
> `https://invoiceapi.informatsoftware.be/swagger/index.html`

GET pdf

Use this endpoint to download the PDF of an invoice for which the "imageUrl" exists.

Request

Headers

Besides the mandatory headers, no other custom headers are required.

URL syntax

https://invoiceapi.informatsoftware.be/api/download/{accessGroupIdentifier}/{invoiceId}/pdf

- accessGroupIdentifier: the access group identifier used for the access token
- invoiceId: the id (guid) of the invoice for which you want to obtain the pdf

Sample

…/api/download/943d5e11-9910-4251-bfce-d1e64c533cc3_502052/9a783289-b3e2-44d9-8ff5-cc7d5460dcae/pdf

Response

The response will be the stream containing the pdf. Content-type will be `application/octet-stream`

***Attention**: Mind that the lifetime of the PDFs is limited. Download the PDFs locally and store them in your system if long-term (+ 1 year) availability is required.*

If the pdf is not (yet) available, status code 400 (Bad request) will be returned.

POST send pdf create command

If “imageUrl” of an invoice is empty / null, you can send a PDF create command using the API. PDF creation is an asynchronous process. Mind that it can take a while before the PDF is available for download.

***Remark**: The PDFs can also be generated from the application iRekeningen by publishing the invoices of type STUDENT to the parent portal using menu Rekeningen > Publiceren > Ouderportaal.*

***Attention:** Mind that Title & Name properties of the invoice address are provided (not NULL!). Otherwise back-end cannot generate the PDF.*

Request

Headers

Besides the mandatory headers, no other custom headers are required.

URL syntax

https://invoiceapi.informatsoftware.be/api/create/{accessGroupIdentifier}/{invoiceId}/pdf

- accessGroupIdentifier: the access group identifier used for the access token
- invoiceId: the id (guid) of the invoice of which you want to generate the pdf

Sample

…/api/create/943d5e11-9910-4251-bfce-d1e64c533cc3_502052/9a783289-b3e2-44d9-8ff5-cc7d5460dcae/pdf

Response

The response will be a status code 202 (Accepted) if the call successfully initiated the creation of the pdf.

When the PDF is created & available on Azure, the “imageUrl” of the invoice is returned by the graphQL request to get invoice data. No notification is sent when PDF is ready.

If the PDF is not available more than 2 minutes after the Create command, initiate a new Create.

Sample code (.NET)

Getting an access token and querying the invoice groups:

```
// ClientId & ClientSecret received from Informat
private const string ClientId = "informat_customer_{clientName}";
private const string ClientSecret = "mySecretCode";

// Scope
private const string Scope1 = "api_informat_invoices.invoices.read.one_of_my_accessgroup_identifiers";

// address to retrieve the access token
private const string IdentityServerBaseAddress = "https://www.identityserver.be";

// invoice api
private const string ApiBaseAddress = "https://invoiceapi.informatsoftware.be";

async Task Main()
{
        var accessToken = await GetToken(Scope1);
        var data = await GetData(accessToken);
        Console.WriteLine(data);
}

private async Task<string> GetToken(string scope)
{
        using var tokenClient = new HttpClient { BaseAddress = new Uri(IdentityServerBaseAddress) };
        var response = await tokenClient.PostAsync("/connect/token", new FormUrlEncodedContent(
                new Dictionary<string, string>
                {
                                        { "client_id", ClientId },
                                        { "client_secret", ClientSecret },
                                        { "grant_type", "client_credentials" },
                                        { "scope", scope }
                }));
        response.EnsureSuccessStatusCode();
        var content = await response.Content.ReadAsStringAsync();
        var json = JsonDocument.Parse(content);
        var token = json.RootElement.GetProperty("access_token").GetString();
        return token;
}

public async Task<string> GetData(string token)
{
        // create HttpClient and use received Bearer token
        using (var apiClient = new HttpClient
        {
                BaseAddress = new Uri(ApiBaseAddress),
                DefaultRequestHeaders = { Authorization = new AuthenticationHeaderValue("Bearer", token) }
        })
        {
                var requestContent = new StringContent("{\"query\":\"query accessGroup($schoolYear: String!,
$accessGroupIdentifier: String!) {\\r\\n  accessGroup(accessGroupIdentifier: $accessGroupIdentifier) {\\r\\n
guid\\r\\n    name\\r\\n    invoiceGroups(schoolYear: $schoolYear) {\\r\\n      id\\r\\n      name\\r\\n
fromDate\\r\\n      toDate\\r\\n      invoiceGroupType\\r\\n      schoolYear\\r\\n    }\\r\\n
}\\r\\n}\\r\\n\",\"variables\":{\"schoolYear\":\"2020-21\",\"accessGroupIdentifier\":\"
one_of_my_accessgroup_identifiers \"}}",
                        Encoding.UTF8, "application/json");
                // call api
                var result = await apiClient.PostAsync("/graphql", requestContent);
                return await result.Content.ReadAsStringAsync();
        }
}
```