INFO-Subscription Documentation

Infosoft AS

Apr 16, 2024

CONTENTS

This is the home of the developer documentation for INFO-Subscription. If you came here looking for the user documentation you are in the wrong place.

If you have no idea what INFO-Subscription is, then head over to the product website and have a look.

CHAPTER

ONE

INTEGRATION OPTIONS

INFO-Subscription is backed by an API that all tenants can make use of.

If you just want the gory details for the API you can browse the Swagger UI or get the OpenAPI/Swagger definition file.

Alternatively, head over to the *Getting Started with the API* section for a more gentle API introduction, or look at our list of *common scenarios*.

Some tenants prefer to use our *managed subscriber experience* which includes

- A self-service portal/web site where subscribers can manage their subscriptions.
- An ordering and user registration site/process where new subscribers can order subscriptions and register a new user.
- An Identity Provider (IdP) solution that contains user credentials and some authorization information for letting users consume their purchased content.

For integrating with the managed subcriber experience header over to our *introduction to the managed experience* .

CHAPTER

TWO

CONTACT INFORMATION AND SUPPORT

Please refer to the section on *Support and Reporting Bugs* for details on how to contact Infosoft.

genindex

2.1 Getting Started with the API

The INFO-Subscription APIs are HTTP-based and designed to somewhat conform to the REST ideology. Purists will note that there are no Hypermedia links, this will may change in the future.

The entire management capability of the platform is exposed in the API, meaning that whatever you see in our supplied management and self-service solution you should be able to do with the API. Meaning you can do things such as:

- Integrate with existing web-shops or frontends
- Provide tailormade self-service for subscribers matching your brand and workflows
- Integrate management capability in a custom business portal, CRM System or similar

Important: Please read the introduction about the terminology before you start banging your head against the wall, it might actually contain a pointer or two that will help you.

In order to actually use the API you will need a couple of things

- A set of client credentials
- A TenantId

The client credentials are used for authentication and authorization to the API which is *described separately*

While the TenantId is used to ascertain which tenant your application is trying to access.

Once you have authenticated, try retrieving a subscriber or all subscriptions to get some instant gratification.

2.1.1 Obtaining Client Credentials

At the current time there is no registration process, so you have to *contact Infosoft* to obtain a set of credentials.

Client Credentials comes in the form of a *client_id* and *client_secret* are used to authenticate (identify) and authorize an application in the API. Together with the `S4-TenantId` (see below), the API will determine if the client should be granted access or not.

Important: The *client_secret* is actually secret. You should keep this safe, and not disclose it to others.

Tip: You should consider obtaining separate credentials for separate applications/solutions.

This way it is easier for you to keep track of the source when investigating issues, errors or perculiar behavior.

2.1.2 Obtaining a Tenant Id

Infosoft should have given you a Tenant Id when you signed up for using INFO-Subscription. If you are a third party, contact the tenant/customer and have them provide you with a Tenant Id. If not please *contact support* for the Tenant Id

Using the Tenant Id

The TenantId is in the form of a UUIDv4/GUID such as `fe923cfe-2e67-4f7a-960a-d4c36fce22c4` and at the current time is required in the `S4-TenantId` header for all requests to the API.

2.1.3 Endpoint(s)

The INFO-Subscription APIs are accessed using https://api.info-subscription.com/ as the base url. All paths, names etc that is referenced here will be relative to that base URL.

2.1.4 Building Requests

In general requests are build such that a HTTP Method/Verb corresponds to a specific type of action.

- `GET` Requests data from a resource
- `POST` Creates a resource or starts a process
- `PUT` Replaces/Completely Updates an existing resource
- `DELETE` Deletes a resource or cancels a process
- `PATCH` Partially updates a resource or state of a process

The APIs expect `JSON` as input and outputs `JSON` unless otherwise noted in the reference documentation.

Tip: Even though the APIs Accept `application/json` as input and responds with `application/json` by default, it is recommended that you set the `Accept` and `Content-Type` headers so there is no doubt for client and server what is requested.

If for some reason you *really really* like XML, Protobuf, Thrift, MessagePack or your own custom content type, then let us know and we might consider implementing it.

2.1.5 Handling Responses

Just as requests are build around the common HTTP Methods, the API responses are constructed around common HTTP Status Codes. Each response contains a code and optionally a body. We strive to follow the same convention throughout the API which should hopefully provide consumers a streamlined experience. Let us know if you discover inconsistencies!

Successful Responses

For succesful requests we generally map the status codes as follows:

- `GET` Returns `HTTP 200 OK` with a body.
- `POST` (i.e. create a new resouce) Returns `HTTP 201 Created`, optionally with a body of the created resource.
- `PUT` Returns `HTTP 204 No Content` with no body
- `DELETE` Returns `HTTP 204 No Content` with no body

You can consider this as a general rule, but to be certain please refer to the reference docs for your particular action.

Error Conditions

There are two categories of errors.

1. Errors that the client can do something about, typically input format or unmet requirements. These produces status codes in the **400-499** range.
2. Errors that is entirely the fault of the server, typically environment errors or implementation errors or other similar unexpected conditions. These produces status codes in the **500-599** range.

Client Errors

As described above these are Errors that the client should handle, typically by displaying some sort of feedback to the user.

Typically client errors are validation/format related and generates a `HTTP 400 Bad Request` response with a validation error instance

Whenever possible multiple validations will be included in a single response, but since some validations are dependant on existance of a given resource this is not always achievable.

The following is an example of a `BadRequest` response with multiple validation failures.

Listing 1: Headers

```
HTTP/1.1 400 Bad Request
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8
Server: Kestrel
Request-Context: appId=cid-v1:ae92dc8e-a862-4e47-92f2-cc707892f433
X-Powered-By: ASP.NET
Date: Mon, 01 Jan 2042 00:13:37 GMT
```

Listing 2: Body

```
{
    "message": "Validation Failed",
    "errors": [
        {
            "field": "Currency",
            "message": "The Currency field is required."
        },
        {
            "field": "ValueDate",
            "message": "Date is out of Range"
        },
        {
            "field": "PaidAmount",
            "message": "The amount is out of Range"
        },
        {
            "field": "PaymentDate",
            "message": "Date is out of Range"
        }
    ]
}
```

Authentication and Authorization errors are strictly speaking also client errors, but the response and the meaning are covered in the *authentication section*

Server Errors

Generally speaking the APIs responds with either `HTTP 500 Internal Server Error` or `HTTP 503 Service Unavailable`.

There might be cases where you get other `5xx` series status codes, but those are always from the hosting environment and thus it is a bit hard to reason about their content in all cases.

HTTP 500 Internal Server Error

These errors should always include a body of the following format

```
{
    "Code": "SOME HTTP StatusCode",
    "Message": "An error message of sorts",
}
```

While the message itself is usually not that informative, we recommend that you log any such errors and open a *bug report* so that we might solve the issue.

HTTP 503 Service Unavailable

Typically waiting a few minutes and trying again should work, if not please open a *bug report* so we can investigate.

Authentication and Authorization Responses

There are currently two auth related responses you can expect and should handle.

- `HTTP 401 Not Authenticated` - Indicates that no authorization information was found, typically because there is no Authorization header or the content of the header was mal-formed.
- `HTTP 403 Forbidden` - Indicates the there was some authorization information, but the resource/action requests requires permissions that the authorized party does not have.

There is currently no body associated with either response, but in case of a `HTTP 401` code a response header `WWW-Authenticate` should be included. An example *401* response:

```
HTTP/1.1 401 Unauthorized
Server: Kestrel
WWW-Authenticate: Bearer error="invalid_token", error_description="The token is expired"
Request-Context: appId=cid-v1:ae92dc8e-a862-4e47-92f2-cc707892f433
X-Powered-By: ASP.NET
Date: Mon, 01 Jan 2042 00:13:37 GMT
Content-Length: 0
```

Tip: While you are most likely to receive these types of responses during development and testing, we recommend you at least handle and log such errors so you have something to debug.

2.1.6 Rate Limiting

At the current time there are no cutom enforced rate limits in place, HOWEVER the underlying platform does enforce some ratelimiting. If you observe these limits, please let us know so we can have a look into what we can do about it.

2.2 What is INFO-Subscription

INFO-Subscription is a service that provides subscription billing and management, in simple terms it creates invoices, manages payments and maintains subscriptions based on these invoices and payments. Most importantly it automates these business processes and attempts to simplify/encode business decisions into these processes, freeing personel to do other things than handling late payments, amount discrepancies and routine dunning in general.

2.2.1 What is it not?

Depending on your point of view it might be easier to understand what it is by describing what it isn't

CRM

INFO-Subscription contains a small ammount personal data, that will allow you to manage simple information about a subscriber. It is not a fully fledged CRM system, neither is it going to attempt to fulfil that role.

However, it is possible to integrate with third party CRMs to synchronize data in a reactive manner, so everyone can bring their own CRM.

Delivery, Distribution and Packaging

Many subscriptions requires some sort of physical distribution of products. While INFO-Subscription is somewhat capable of producing information to distribution and packaging systems, it is not capable of generating package slips, driver routes etc.

Magic

No, unfortunately it is not powered by magic just yet, but we have contacted Dr Strange in an attempt to get a Powered By Magic integration.

2.2.2 What INFO-Subscription does

So that was a list of things INFO-Subscription does not do, now then what does it actually do. While not a complete feature list it gives you an overview of what you can expect. For feature information refer to the product website

Subscription Management and Billing

INFO-Subscription provides capabilities to:

- Create new subscriptions.
- Handle cancelations and upgrades/downgrades subscriptions (both planned and immediate adjustments).
- Customize billing cycles to suit various business needs.
- Automatic proration for upgrades/downgrades.
- Automated handling of late payment and amount discrepancies.

All of this is done by codifying business requirements/decisions into configuration and setup of various rules.

In this case billing is the process of figuring out when and which subscribers should receive an invoice, and produce said invoices based on the details of the purchased plan/product. Additionally billing refers to handling of payments and making sure that subscriptions are cancelled on missing payments etc.

Automatic Payments

Part of the billing engine also manages automatic payments with cards, direct debit and similar, making sure that most subscribers pay the right amount at the right time.

Automated Dunning

Sometimes subscribers neglects to pay their invoices, in these cases INFO-Subscription makes it possible to automate dunning procedure to generate reminders, and adding handling fees etc.

2.3 Terminology and Definitions

Subscriber
Account Owner
An entity who owns a given subscription is considered to be legally responsible for the subscription. All financial transactions for a subscription is associated with the subscriber (in the general case).

End User
User
A person/entity who interacts with INFO-Subscription either as a tenant/merchant or as a potential subscriber.

Subscription
An agreement for an organization to provide/deliver a service or product to a subscriber in a given period of time. A subscription is typically associated with a specific set of rules, called a plan, which describes the particular terms of the subscription. See *plan* for details.

Plan
Package
A set of rules that governs the details of a subscription, such as the duration, continuation, billing and fees.

Organization
A legal entity, company or similar who offers one or more products as a subscription.

Payment
A Payment

Subscriber Account
An account within an organization that belongs to a subscriber

Product
A product is something that is sold/purchased either as a service, benefit or a physical wares that can be subscribed to

Payment Agreement
An agreement on a means of payment/billing between an organization, a subscriber and optionally a third party that allows the organization to obtain payments from the subscriber.

Payment Service Provider
PSP
From wikipedia : A payment service provider, offers shops online services for accepting electronic payments by a variety of payment methods including credit card, bank-based payments such as direct debit, bank transfer, and real-time bank transfer based on online banking. Typically, they use a software as a service model and form a single payment gateway for their clients (merchants) to multiple payment methods.

Mastercard Payment Solutions (formerly NETS)
Bankgirot
SwedbankPay
Vipps Company
A specific PSP - see *PSP*

eFaktura
eFaktura is a Norwegian electronic invoicing method for invoicng through most Norwegian online banking soltions provided by Mastercard Payment Srvices. May be combined with AvtaleGiro *AvtaleGiro*

Vipps
Vipps is a Norwegian App for payments managed by mobile phone, either through payment cards or direct bank account transfers behind the scenes.

AvtaleGiro
AvtaleGiro is the direct debit solution in Norway, handling recurring payments with little customer interaction after setup. It is provided by Mastercard Payment Solutions. May be combined with *eFaktura*.

Autogiro
AutoGiro is the direct debit solution in Sweden, handling recurring payments with little customer interaction after setup. It is provided by *Bankgirot*.

2.4 API Authentication and Authorization

INFO-Subscription APIs uses OAuth2 and OIDC for API Authentication. OAuth2 knowledge on its own is sufficient but the metadata discovery of OIDC makes the process simpler for most libraries.

The underlying authentication service is provided by ADB2C the following should get you started, if you have questions reach out to support.

Important: This sections describes authentication for the API, look to the *User Authorisation Quick Start* for details on end-user/subscriber authentication and authorisation.

2.4.1 Minimum Required Details

The quick'n'dirty details for those already familiar with OAuth2 or OIDC flows who just wants to get started.

- Token URL: at https://prodlogins4.b2clogin.com/prodlogins4.onmicrosoft.com/B2C_1A_V2SIGNIN/oauth2/v2.0/token
- Grant Type: Typically `client_credentials`
- Audience: 7bdcdd56-3ba6-4e6d-a841-db4816d7909d
- Metadata Url: https://prodlogins4.b2clogin.com/prodlogins4.onmicrosoft.com/B2C_1A_V2SIGNIN/v2.0/.well-known/openid-configuration
- Scope: https://prodlogins4.onmicrosoft.com/api/.default

Remember to *Reuse the Token* until it expires!

2.4.2 Token Acquisition

If you are somewhat familiar with OAuth2 and OIDC terminology we rely on ADB2C as the *Authorization server*. If you have no clue what an authorisation server is, it is basically the thing that identifies the application for the user and the API.

To access the API you should obtain a token using for instance the `client_credentials` grant type.

To obtain a token `POST` a request to https://prodlogins4.b2clogin.com/prodlogins4.onmicrosoft.com/B2C_1A_V2SIGNIN/oauth2/v2.0/token with a body containing your chosen flow type, client credentials and the audience.

curl

http

```
curl --request POST \
    --url 'https://prodlogins4.b2clogin.com/prodlogins4.onmicrosoft.com/B2C_1A_V2SIGNIN/
↪oauth2/v2.0/token' \
    --header 'content-type: application/x-www-form-urlencoded' \
    --data 'grant_type=client_credentials&client_id=YOUR_CLIENT_ID&client_secret=YOUR_
↪CLIENT_SECRET&scope=ADB2CAPISCOPE'
```

```
POST https://prodlogins4.b2clogin.com/prodlogins4.onmicrosoft.com/B2C_1A_V2SIGNIN/oauth2/
↪v2.0/token HTTP/1.1
Host: prodlogins4.b2clogin.com
Content-Type: application/x-www-form-urlencoded
Accept: application/json

grant_type=client_credentials&client_id=YOUR_CLIENT_ID&client_secret=YOUR_CLIENT_SECRET&
↪scope=ADB2CAPISCOPE
```

The response comes in the form of a JWT token, a type, and an expiration time relative to the issue time.

```
{
    "access_token":
↪"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Imhxb2k2Z19yanhEdmtuWmVKNDVIWk9UbnhkeHNrSnVldTRCdE04ajRQ
↪eyJpc3MiOiJodHRwczovL3FhbG9naW5zNC5iMmNsb2dpbi5jb20vZDgxODM4NGEtZjZjZS00ZDJhLWIwNWEtMjI5NTUxZjY4YTJhL
↪Y2syarjQxh1Bxc6W9vsnXo6TvOgFZZGb4cy21wZNzHy6s31XmmuUtQ-Aae7CODXYhgLtE5PEmcgGzBSlxVbW_
↪MM937e1yx6y0fDgr9wONcFc5RRxjYZctXVOH38TMWj2p-LSOSGZPgVH_CD1vEsA05urszP4Ql1kQwhHY5RN7Y-
↪1PI-KupPkIocddJchl3jw_HB8_
↪iIP9IjUhxptH6bAXjDwnzoVbMVVfCy4qMpf4EpxsbOP9nUdyhDJBHnTmYamnOKEk6JfDMJ1Y_
↪f2seLK9HZ7GVT2RL7hn1X7zgL474LgOVOwOk8VolTzHOxV2kmacAVUlL0X6x0mfpAEEh9j8Q",
    "token_type": "Bearer",
    "not_before": 1664289386,
    "expires_in": 3600,
    "expires_on": 1664292986,
    "resource": "726389bb-5f80-47e8-a115-9427119f9299"
}
```

If you want to look at whats inside the token go to https://jwt.io or https://jwt.ms and copy/paste the token to have a look.

2.4.3 Using the token

Once you have a token, it should be included in the `Authorization` header as a `Bearer` token. It should look something like the following

```
Authorization: Bearer␣
↪eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Imhxb2k2Z19yanhEdmtuWmVKNDVIWk9UbnhkeHNrSnVldTRCdE04ajRQT
↪eyJpc3MiOiJodHRwczovL3FhbG9naW5zNC5iMmNsb2dpbi5jb20vZDgxODM4NGEtZjZjZS00ZDJhLWIwNWEtMjI5NTUxZjY4YTJhL
↪Y2syarjQxh1Bxc6W9vsnXo6TvOgFZZGb4cy21wZNzHy6s3lXmmuUtQ-Aae7CODXYhgLtE5PEmcgGzBSlxVbW_
↪MM937e1yx6y0fDgr9wONcFc5RRxjYZctXVOH38TMWj2p-LSOSGZPgVH_CD1vEsA05urszP4QllkQwhHY5RN7Y-
↪1PI-KupPkIocddJchl3jw_HB8_
↪iIP9IjUhxptH6bAXjDwnzoVbMVVfCy4qMpf4EpxsbOP9nUdyhDJBHnTmYamnOKEk6JfDMJ1Y_
↪f2seLK9HZ7GVT2RL7hn1X7zgL474LgOVOwOk8VolTzHOxV2kmacAVUlL0X6x0mfpAEEh9j8Q
```

If the client that obtained the token is allowed to communicate with the API, the request will go through, and otherwise you will get an `403 Forbidden` error. Requests with invalid tokens, or without tokens will be presented with a `HTTP 401` error.

2.4.4 Authorisation Flows or Grant Types

There are multiple ways to authorise an application to work with an API, in OAuth2 lingo these different flows are called grant types. Depending on your use case, you may use the flow you are most comfortable with.

For machine to machine interaction, with no end-users involved, the recommended option is to use *client_credentials* grant type.

For cases with interactive administrative/merchant users, the recommended option is to use *implicit* or *authorization_code* grant types.

2.4.5 Reuse the Token

All API tokens contains information about when they expire relative to the issuing time. That means a token can be re-used for all machine to machine requests in the timeframe for the given client/audience/tenant pair. For most integrations that means only one token is needed at a time. For some integrations it may require a few more, but a limited set of tokens needs to be generated and they can be kept untill they expire.

While obtaining a new token is a relatively quick action, the token should be re-used until it expires for two primary reasons:

1. It improves the performance/experience for the integration i.e. not every API call needs a new token authorization, but just once every hour/day the overhead is paid.
2. Costs, there is an indirect cost associated with API Authentications, in case of repeated misuse extra charges may be added to the tenant.

2.4.6 Legacy Token Acquisition using Auth0

The following describes obtaining tokens using Auth0 as the provider. This solution is now deprecated, but it still works for tenants that were using it previously. The documentation is kept for reference as long as there are active tenants running with this setup.

The service is provided by Auth0, and thus the gory details may be found at the Auth0 website, but the following should get you started.

Short Version

The quick'n'dirty details for those already familiar with OAuth2 flows.

- Token URL: at https://infosubscription.eu.auth0.com/oauth/token
- Grant Type: Typically `client_credentials` but any of the common grant types works. Contact support for specific grant types.
- Audience: https://api.info-subscription.com/

Remember to re-use the token untill it expires!

Authorisation in detail

To access the API you should obtain a token using for instance the `client_credentials` grant type.

To obtain a token `POST` a request to https://infosubscription.eu.auth0.com/oauth/token with a body containing your chosen flow type, client credentials and the audience

```
{
    "client_id" : "{{client_id}}",
    "client_secret":"{{client_secret}}",
    "audience":"https://api.info-subscription.com/",
    "grant_type":"client_credentials"
}
```

The response comes in the form of a JWT token, a type, and an expiration time relative to the issue time.

```
{
    "access_token":
↪"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Ik1FVTJOa1pEUWpCRVFUWkNRamxCTTBaRFJUZ3pRVEZHUXpaQ09VVTRN
↪eyJpc3MiOiJodHRwczovL2luZm9zdWJzY3JpcHRpb24uZXUuYXV0aDAuY29tLyIsInN1YiI6Ik5Cc2gyalFVbTE2NXNBWTVmZWQzR
↪fLiToHzpMzcDkBarLu9MYR-LTYR4V0MCeoG4_sEhoH4ykDu0lhp-
↪cgloJnYR6jEFNcK6u8difFViVSrrAnM7QPCp2eqptZQxkqjX0ZNdNUbkvSnpL7iFHKkEvy7DdRLjHkX6oJq_
↪Le9ww6fKmdhVqvEnbu8h39mMWQPHGk0dh0mketr6tZRxu0WGBYusbeZOH9lkn3mQhAFl1nzqE3sePjTkwe1rah8FKHQhI2xYfd-
↪dwMWAiPiXLRS_H5l9NyjtdcIvtXLnfWTM_eo0qAHPh1Q_4TlEPFptLk37Bx3NE6U5UM9EiQLUP0jdxOr9_
↪2mPST70bIKQxh60YRgOWd8Jug",
    "expires_in": 86400,
    "token_type": "Bearer"
}
```

Once you have a token, it can be used as described in *Using the token*

2.4.7 Authentication Libaries and SDKs

Most OIDC/OAuth2 conformant libraries and SDKs can be used to obtain tokens.

Our recommendation is to use a library if at all possible instead of relying on a hand rolled solution.

Most libaries will do, but since we have used Microsoft ADB2C and Auth0 as the token providers those are the libraries we have tested to some degree.

Microsoft Authentication Library - MSAL

Microsoft provides a range of client SDKs that are designed to work with the Microsoft Identity Platform on a wide array of languages and platforms.

It supports the various quirks of ADB2C, but should work for most OIDC providers.

Most of the SDKs includes transparent in-memory caching of tokens, and in some cases extension libraries for serialized token caches exists as well.

Auth0 Authentication and Management Libraries

Similar Auth0 provides a wide range of client SDKs that are easily consumed and supports the various Auth0 quirks.

There are event more languages and frameworks supported for Auth0 than MSAL. Similar to the Microsoft libraries, some of these provides in-memory caching as well as serialized token caches.

2.5 Common Scenarios

Some integration scenarios are more common than others. The following is a short list of scenarios that we have identified as common and worth highlighting.

- *Creating New Subscriptions*
- *Upgrading and Downgrading Subscriptions*
- *Authentication and Authorizing Subscribers*
- *Managing Subscriber Payment Agreements*

Is your desired scenario missing? It may still be found somewhere within the documentation.

Let us know if you can't find it and we will direct you, or if the documentation is lacking we will attempt to rectify it.

2.6 Managed Subcriber Experience

The managed subscriber experience is a turn key solution for tenants that do not want to build their own customer facing solutions.

It consists of three parts which works in tandeem to provide a complete solution for end users (subscribers).

- An Identity Provider (IdP) utilized for user authentication (username/password, social accounts etc).
- A salesprocess that takes care of registering new users, subscribers and subscriptions complete with payment agreements, invoice contact details.
- A self-service management site where users can see and manage their subscriptions, change payment agreements and get an overview of their invoices.

While the components provided are enough to get most business going, there is typically some requirement to make them interact with an external web-site. Either to funnel users into different sales processes (different plans, campaigns, products etc), or to authenticate and authorize users for consuming their content.

A typical site integration consist of the following steps:

1. Add a link to the self-service site (see below).
2. Construct a landing page for common product/subscription offerings redirecting to the right *sales poster configuration*.
3. Implement *authentication and authorization* for protected resources.

2.6.1 Accessing Self-Service/Subscription Management

Once an end-user/subscriber has made a purchase and created a user, regardless of how that was achieved, the user may need to manage the subscription or just see the latest invoice.

The self-service management solution can be access by navigating to https://{tenantName}-s4.azurewebsites.net where *tenantName* is the name of the tenant.

If the tenant is configured with a *custom domain* that domain may be used instead.

2.7 Orders and User Registration

INFO-Subscription contains a turnkey Salesposter/Ordering solution that can get organizations up and running in no time.

For tenants where the managed subscriber experience is enabled, the salesposter is available on a URL following the template: https://{tenantName}-s4.azurewebsites.net/salesposter, where *tenantName* is the name of the tenant.

If the tenant is configured with a *custom domain* that domain may be used instead of the azurewebsites subdomain, such as https://subscription.example.com/salesposter

Creating new subscriptions is as simple as providing the user with a link to the salesposter. Afterwards a user will exist that can be *authenticated and authorized* by external parties, and recurring billing will follow automatically until the subscription is cancelled.

Try it out using our experimentation site https://experimentation-s4.azurewebsites.net/salesposter

2.7.1 Configuration Parameters for the ordering process

There are two levels/areas of configuration for the salesposter.

A set of configuration options are static i.e. it does not vary from sale to sale or subscriber to subscriber, and is configured during onboarding and in some cases via the Merchant interface of INFO-Subscription. These are things covered by the user documentation and as such are not relevant from a technical/integration standpoint.

Another set of configurations are dynamic and varies from process to process. These options can be changed from session to session at will.

The Salesposter primarily uses URL query parameters to control presentation and validation logic.

Once a user follows the link with query parameters, they are copied to an encrypted cookie and stored throughout the ordering process. To change these params, simply make a new URL with different values.

The following table outlines the various parameters, some have a set of fixed possible values.

Parameter Name	Description	Default Value	Possible values
lang	Defines the display language	nor (Norwegian)	*Available Languages*
redirect	A redirection location for when the order has completed		Any URL
pty	A list of available payment types	1 (Swedbank Pay)	*Payment Types*
loginWith	Defines which user credential providers to expose	internal	*Credential Providers*
sourceUrl	An encoded URL that will be sent to **Kilkaya** for tracking	Off	Any encoded string
templatepackageid	Subscription Plan Id being ordered	Custom	Any valid Template Plan
organization_id	The Organization the order should be placed on	Custom	Any valid organization
adr	Controls the address section	On	*Mandatory Toggle Options*
tel	Controls the phone section	On	*Mandatory Toggle Options*
org	Controls the organization number section	On	*Mandatory Toggle Options*
inv	Controls the invoice Contact section	Off	*Mandatory Toggle Options*
invtext	Header text for invoice contact section	Language dependent	
orginv	Controls additional organizaiton number section	Off	*Mandatory Toggle Options*
co	Controls the co section	Off	*Mandatory Toggle Options*
invref	Controls primary contact buyer reference section	Off	*Mandatory Toggle Options*
invpayref	Controls invoice contact buyer reference section	Off	*Mandatory Toggle Options*

The following is a series of options tables (referenced by each parameter above).

Table 1: Language Options

Value	Description
eng	English
nor	Norwegian
swe	Swedish
sam	Northern Sami

Table 2: Payment Type Options

Value	Description
1	Swedbank Pay
2	Vipps
9	Invoice

Table 3: Credential Providers

Value	Description
Internal	Uses internal login mechanism (“local” users)
Google	Google accounts
Facebook	Facebook accounts

Table 4: Optional/Mandatory Toggling Options

Value	Description
On	Shows the section
Off	Disables/Hides the section
Required	Shows the section and makes it mandatory to fill

Simple Example

The following is an example of how the Salesposter may be configured to do a specific thing. In this case the configuration does the following:

- Restricts the Payment Type to be Swedbank Pay
- Overrides the default Subscription Plan
- Uses a *custom domain* (not really salesposter specific).

Example Url

https://experimentation.minside.info-subscription.com/salesposter?pty=1&templatepackageid=14714f54-dbf4-4899-a9c7-51763d536568

2.8 Authentication and Authorization Quick Start

For many subscription services, it is a requirement that subscribers can sign-in and access their subscription. Either to consume it or to manage it, and often both.

INFO-Subscription provides optional, built-in, components to solve common authentication and authorization needs. The components are built such that they can be used with any identity provider (IdP), but with a specific common authorization means. For advanced authorization scenarios where multiple systems contribute to the authorization flow, the components may be used partially or skipped altogether depending on the requirements.

The following is a quick-start to help get you started if all you require is a simple option to authenticate and authorize subscribers to access one or more specific products.

The default subscriber authentication and authorization component is built around Microsoft Azure ADB2C (henceforth just ADB2C). Its a long winded name, but it is basically just an Identity Provider for consumer/customer accounts, in this case INFO-Subscription subscribers.

ADB2C is an Identiy Provider (IdP) and out of the box provides Authentication using OAuth2 and Open Id Connect (OIDC). As part of the authentication flow INFO-Subscription extends the identity with a set of attributes that may be used to Authorize the user for accessing a given product.

Since ADB2C uses OIDC protocol it is likely that the stack you are using has a component or library that can help you along.

The official ADB2C documentation contains a guide on how to send authenticatation requests.

The guide will refer to variables to replace such as *tenant*, *appId*, *policy* and more. Refer to the table below on how to obtain all of this information, when you don't have access to the desired information.

Table 5: ADB2C Reference variables

tenant	Obtained when signing up an application. Typically the INFO-Subscription tenant name appended with *S4Prod* (Example: *experimentationS4prod*)
policy	Defaults to *B2C_1_SignIn*
client_id	Obtained when signing up an application
client_seci	Obtained when signing up an application
redi-rect_uri	Configured when signing up an application, this is your application URL provided when you signed up the application.

There are quite a few parameters in the table for the *auhtorize* endpoint, most of which is something you need to consider if hand building the request. If you are using a component or library a lot of this is probably taken care of out of the box.

2.8.1 Example Authorization Code Flow

The following are exsamples of an authorization code flow, to get you started, but we recommend learning about Open Id Connect and specifically ADB2C to get the most out of your authentication and authorization process.

Note: If you have a test account on the Experimentation tenant, the following should be reproducable using just your browser.

Obtaining the authorization code and id token

```
GET https://experimentationS4prod.b2clogin.com/experimentationS4prod.onmicrosoft.com/b2c_
↪1_signin/oauth2/v2.0/authorize?client_id=1b162230-180c-4648-9d0f-a313bb86510c&response_
↪type=code+id_token&redirect_uri=https%3A%2F%2Fjwt.ms%2F&scope=openid%20offline_access&
↪response_mode=fragment HTTP/1.1
Host: experimentationS4prod.b2clogin.com
Connection: keep-alive
Accept: */*
```

Once the login prompt has been taken care of, a redirect should occur to something like

Listing 3: Sample Redirect/Response

```
GET https://jwt.ms#id_token=eyJ0XAi...4faeEoQ&code=XfTTAAAvPM1KaPlrEqdFSB..&state=the_
↪state_you_sent_in_during_step_one HTTP/1.1
Host: jwt.ms
Connection: keep-alive
Accept: */*
```

At this point you can either use the *code* to obtain an *access_token* or you can verify and decode the *id_token*

Verifying and Decoding the token

The ID Token provided is a `JWT` token, encoded and signed by the IdP. The signature can be verified by using the key provided in the OIDC Metadata endpoint. Verification is outside the scope of this documentation, we recommend you entrust that to someone who knows cryptography (i.e. a library/component).

Once decoded the `id_token` contains the identification of the user as well as his/her *SubscriberId* and any *Products* that are accessible currently.

The two claims containing the identification:

- `extension_SubscriberId` - A SubscriberId representing a specific subscriber.
- `extension_Products` - A JSON string containing an array of products and the users current validity of those products.

Listing 4: Sample Id Token

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Ilg1ZVhrNHh5b2pORnVtMWtsMll0djhkbE5QNC1jNTdkTzZRR1RWQndhTms
↪eyJleHAiOjE2Njk5NzcwOTgsIm5iZiI6MTY2OTk3MzQ5OCwidmVyIjoiMS4wIiwiaXNzIjoiaHR0cHM6Ly9leHBlcmltZW50YXRpb
↪ADc-WY-w0CH3yY6Ch97vox2F-xEIQOsb3jKb45VJjfwKNuWWGauO5zsdwLyBQk0w_
↪HVU1738SsFYfO0Iwe4cXM6gpf5LVwc8vjugD6pBZLN6SLPEnSCoMdDwRGT_hN_qaotZD7mKTXV3H0io98-
↪btGWs9zjkzLX3APteDih7P8Kn-AjZPVxwnVgkRI0w3-SjkU1fiAtlRYYueFA_
↪0pIEHeLfF9TcvknPvFQry8gvJZbm5-QRgP2z4n6_Hvh9prPDtot2BKahKkgCV877K4U5cYJkWWO8a0LVgKXbx0-
↪7uI9YnjdKU1Hmloh7H70GpaxTfOoxzJxcs3Q9x5MafJvMdg
```

Listing 5: Sample Id Token Decoded

```
{
    "exp": 1669977098,
    "nbf": 1669973498,
    "ver": "1.0",
    "iss": "https://experimentations4prod.b2clogin.com/b2f8feca-1a5c-4090-ab41-
↪9013d3420118/v2.0/",
    "sub": "62a786d2-5cd2-4a26-9cb0-18b056b9562f",
    "aud": "1b162230-180c-4648-9d0f-a313bb86510c",
    "iat": 1669973498,
    "auth_time": 1669973498,
    "name": "esbbach+testuser1@infosoft.no",
    "emails": [
        "esbbach+testuser1@infosoft.no"
    ],
    "extension_SubscriberId": "73265483-0a64-4acc-9ccf-11359ef5ce9f",
    "extension_Products": "[{\"Id\":\"70e75bc0-6c3d-4934-a2e8-08d80b92b721\",\"ValidFrom\
↪":\"2022-06-29T05:37:25.8669071+00:00\",\"ValidTo\":\"2023-06-29T05:37:24.
↪8669071+00:00\"}]",
    "tfp": "B2C_1_Signin",
    "c_hash": "flpwPutr0z_iuOTww7zJdA"
}
```

The above JWT is a sample, not all fields will always be available or have values.

`extension_SubscriberId` may be empty for family members and other shared users.

`extension_Products` can be empty whenever no mapping exists that grants product access.

Important: ADB2C has some perculiar behaviour when the user has forgotten the password. Refer to the *forgotten* *password* section for the details on how to handle it.

2.8.2 Authorization Alternatives

Now that a token has been retrieved and decoded, the next step is to do the actual authorization.

Depending on your need there are two simple alternatives that may be used, either:

1. Verify that there is something in the `extensions_Products` list.
2. Verify that the product requested is available in the `extensions_Products` list.

Note: `extensions_SubscriberId` is not part of the suggested auhtorization process, as in most general cases it is not needed.

Important: The two authorization/verification alternatives are suggestions, verify with the business requirements that they will suffice for your case.

2.9 Microsoft ADB2C

The following contains a few pointers related specifically to interactions with ADB2C that integrations should consider in the interaction. For other customization and interaction options refer to the official ADB2C product documentation.

2.9.1 Handling Forgotten Password

With the INFO-Subscription provisioned/managed ADB2C instance configuration, some care is required for handling session that starts by the user resetting the password.

The scenario/flow is roughly as follows:

1. The user is presented with a login prompt.
2. The users clicks “Forgot Password”.
3. User goes through email verification and enters new password (and a confirmation).
4. The user is logged in without having to enter his/her credentials after the change.

The issue is that during step 4, the regular server side login process is NOT executed.

The end result is that the id_token generated does not include the *extension_SubscriberId* nor *extesion_Products*.

Recommend Workaround

The recommended workaround for this scenario is for the integration application to handle the password journey by forcing a new login.

1. Detect that token contains the property *isForgotPassword* with a value *true*.
2. Clear local session cookies.
3. Re-run login flow.

The result will be that the user needs to enter his/her credentials again, and the resulting token will be populated with the correct information.

The above approach is implemented in the INFO-Subscription self-service application.

Listing 6: Sample Decoded Id Token for the Forgotten Password Journey

```
{
    "ver": "1.0",
    "iss": "https://experimentations4prod.b2clogin.com/b2f8feca-1a5c-4090-ab41-
↪9013d3420118/v2.0/",
    "sub": "62a786d2-5cd2-4a26-9cb0-18b056b9562f",
    "aud": "1b162230-180c-4648-9d0f-a313bb86510c",
    "exp": 1707321390,
    "nonce": "defaultNonce",
    "iat": 1707317790,
    "auth_time": 1707317790,
    "isForgotPassword": true,
    "name": "esbbach+testuser1@infosoft.no",
    "emails": [
    "esbbach+testuser1@infosoft.no"
    ],
    "oid": "62a786d2-5cd2-4a26-9cb0-18b056b9562f",
    "tfp": "B2C_1_Signin",
    "nbf": 1707317790
}
```

2.9.2 Single Sign On

By default all INFO-Subscription provisioned ADB2C tenants are configured with SSO being enabled across the tenant.

In essence that means, that if a user is signed in to one application, for instance self-service, and have “Keep Me Signed In” toggled on, there should be no login prompt but instead he/she should be automatically signed in.

There is one requirement in that the login request should NOT include the query parameter *prompt=login*, as soon as that is included the SSO session is terminated.

2.10 Product Authorization In Depth

The following describes the relationship between users, subscribers, authentications and authorizations and how these concepts are treated by INFO-Subscription.

Authentication and authorization is often mixed together, because they often follow right after each other. For good measure the following is how the terms are defined throughout the documentation:

- Authentication is the process of identifying the user is who she says she is, typically handled by asking for a password and other credentials.
- Authorization is the process of determining what the user should have access to do. I.e *Is Jane Doe allowed to read the sports section?*

INFO-Subscription does not make any assumptions, restrictions or requirements about how tenants want to Authenticate and Authorize. Meaning each tenant is free to use their own IdP and roll their own custom authorization process.

2.10.1 Authorization Building Blocks

The user authorization module provides building blocks which can be used in different ways to help achieve a simpler, faster and more robust auhtorization process.

There are several central parts

- Identity Providers
- Users
- Authorization Sources

The following goes into details about each area in turn, describing how they work, can be used by external parties, and how INFO-Subscription use these for providing an out-of-the-box auhtorizaiton experience using ADB2C and Auth0

To create, update and otherwise manipulate these items look at the API Reference documentation, specifically anything under the */authorization* path.

2.10.2 Identity Providers

In short: An identity provider represents a source of users.

An identity provider can be a database, an online service, a location or anything for that matter. Each user can only exist in one provider at a time, though there is nothing that restricts you from having multiple providers and having the same user added multiple times, one for each.

The fact that an identiy provider can be almost anything is used as the background for solving the special case of Site-Access Authorization.

There are two built-in identity provider types that INFO-Subscription knows about and will offer to do some automation for, but even though you utilize one of these types, the automation is entirely optional.

- ADB2C
- Auth0

For both of these built in providers the support revolves around two things:

- Generating claims data in the *id_token* and *access_token*
- Managing users from the self-service and Merchant clients

Both cases needs to make SOME assumptions and requirements on how authorization works, but as mentioned, this part is optional to use.

2.10.3 Users

A User resides with an Identity Provider and exists primarily as a mapping construct between Identity Providers, Subscribers and Products.

A user has the following primary properties:

- `ExternalId` - The Id as it is on the Identity Provider. Useful for quick look up after authentication.
- `SubscriberId` - An optional Id that maps the user to a specific subscriber.
- `IdentityProviderId` - The Id that shows which identity provider this belongs to. Only useful for multi-IdP scenarios.

2.10.4 Authorization Sources

Some businesses need little more than a *user* and an *identity provider* mapping the user to a susbcriber to solve their needs. Others require a bit more magic to make things working, this is where **Auhtorization Sources** come into play. Authorization sources are the auto-magic of the user authorization that INFO-Subscription offers.

Important: An essential assumption is that an authorization is considered to be access to one or more products. If this is NOT true for your business, the automation offered is not suitable for your use case.

It might still be the mapping support used to generate the product list is relevant, but not the end result.

An Authorization Source is a mapping between a user, and an entity that INFO-Subscription knows about.

Each source grants access to one or more products depending on the source.

Whenever an authorization source is added or removed from a user INFO-Subscription will attempt to resolve the list of products that the user has access to based on this source.

At the time of writing the following sources exists, each with its own logic for mapping products onto the users.

- Subscriber
- Subscriber Account
- External Static Sources

Adding an authorization source to a user will result in a list of products being built and associated with the given user.

Each addition, modification or removal will cause a rebuild of the entire list, so please allow a few seconds before the list is completely up to date.

Subscriber Authorization Source

The subscriber authorization source can be added either directly, or by mapping the user to a subscriber. It is the only source that can be implicitly mapped.

The authorization source will enable all products for which the *subscriberId* owns a subscription that is currently valid.

A valid subscription in this context is a subscription that:

- Is started before now
- Is ending later than now
- Is not cancelled currently

Whenever a subscription is created, renewed or cancelled all users connected to a subscriber via the subscriber authorization source will have its product list updated.

This is the mapping type used by INFO-Subscription for the main/primary user account on a subscriber. The self-service uses this to implicitly indicate ownership.

Subscriber Account Authorization Source

Adding a Subscriber Account Id to a user will do something similar to to the Subscriber Authorization Source.

The source will enable all products for which the *subscriberAccountId* owns a subscription that is currently valid (using the same validity conditions as the subscriber source).

This is the mapping type used by INFO-Subscription for adding shared and family users.

A user can be mapped to a subscriber and multiple subscriber accounts at the same time that have nothing to do with each other.

External Static Auhtorization Source

In some scenarios it may be something external that grants access to a specific product.

Consider a case where a timed purchase in an external system should temporarily grant access to the same products as a subscription does. It could be a timed coupon for instance.

You could either build your content system around being able to read these time based coupons or you could let some middleware inject the Product into the user and use the regular authorization routine.

An *External Static Authorization Source* is just that: Something external, and something static (from the point of view of INFO-Subscription).

All products here are added/removed as is. Only adding or removing a source will cause the product to be enabled or disabled. There is nothing inside INFO-Subscription that will manipulate this list automatically.

2.10.5 Accessing Product Authorizations

Based on the list of authorization sources, a list of products is built for the user. That list is collapsed so duplicates are removed.

The list can then be access directly by the external user id provided upon registration using the endpoint `/authorizations/identityprovider/{identityProviderId}/user/{externalId}`

Enriching Tokens During Sign-in

Most online IdP solutions allows for some sort of external API connections during sign-in. If mapping is done correctly, these sign-in procedures can use their own User Id to obtain a list of products and make decisions on what to put in the token.

For tenants running the INFO-Subscription provisioned ADB2C the `id_token` is populated with subscriberId and products during sign-in as described in the quick start.

The same can be achieved if you are running your own ADB2C instance. In that case please *contact support* as configuring it is a bit outside the scope of this documentation.

Note: We are working on migrating our Auth0 based tenants to a solution where the tokens are enriched in the same manner, but it requires some work and most importantly coordination.

If you are interested in transitioning to this new solution please reach out to *support*.

2.10.6 Scenario: Site Access

For some business scenarios, it is common to provide access to a product for a given site or location.

A common example is scientific journals, a university buys access and all employees and students are granted access whenever they are on campus.

There are many solutions for this problem, but a common solution is to somehow map the IP Address of the campus, and all requests originating from that IP will be allowed access.

The model provided by INFO-Subscription can help automate a solution for this scenario, but some custom authorization logic is still required to make it work.

The scenario assumes the following:

- A subscription for a given product is required to access content.
- A campus/site/location can be associated with one or more public IP Addresses.
- The content management system can identify the accessing IP.

To achieve such a site access authorisation a few things should be done.

Using INFO-Subscription do the following:

1. Create a Subscription Plan with the required product(s)
2. Create a Subscriber and a Subscription on the new plan. If you do this via the API, the Sales Poster or the Merchant is not important.
3. Record the Subscriber Account Id.
4. Create an Identity Provider and call it `Site Access` or something similar.
5. Add a new user for the `Site Access` provider, in the `ExternalId` note the IP Address of the site.

6. Create a `SubscriberAccountSource` using the just created `userId` and the `subscriberAccountId`.
7. Test it by providing the IP Address to the endpoint `/authorizations/identityprovider/{identityProviderId}/user/127.0.0.1`

In the CMS do the following:

1. Before Authentication obtain the IP address and look it up using the endpoint from above.
2. If a valid/success response is given grant access, otherwise start the regular authentication and authorisation process for users.

Now everyone on campus has access without authenticating and without an individual subscription.

If the subscription is upgraded, everyone gets access to the upgrade. Should the subscription be cancelled, everyone looses their access.

2.11 Legacy Authentication Using Auth0

The following section is here to document the legacy authentication process for tenants using Auth0 for their users.

2.11.1 Authenticating Subscribers

INFO-Subscription has a built-in module utilizing Auth0 for managing subscriber users, as such developers familiar with Auth0 can use what they know of Auth0 out of the box, and may refer to their documentation for details.

In order to authenticate users, a regular Open Id Connect (OIDC) login flow may be used.

For most scenarios we recommend using the Auth0 Universal Login for authentication, Auth0 provides a series of Quick Start guides for different application types at https://auth0.com/docs/quickstarts.

The Auth0 guides will refer to the admin section for various details such as *domain*, *connection*, *client_id*, *client_secret* and more. Refer to the table below on how to obtain all of this information, since you do not have access to the Auth0 admin section.

Table 6: Auth0 References

Connection	<Your Tenant Name>
Domain	infosubscription.eu.auth0.com
Client_Id	Obtained when signing up, or by contacting infosoft
Client_Secret	Obtained when signing up, or by contacting infosoft
Callback	Configured by infosoft upon request
Logout URL	Configured by infosoft upon request

Note: It is on our backlog to allow for more administrative control of these settings, but currently settings like these needs to be coordinated with Infosoft.

Tip: You may also refer to the *API Authentication and Authorization* section, but keep in mind you need to use an interactive/on-behalf-of grant type and not the *client_credentials* grant type.

Keep in mind that these are two independent sections of the application with different sets of users, you should treat the authentication as two separate things as well (i.e. one for admins/machines and another for subscribers).

There are a few customizable options for the Login Page, refer to the section *Login Page Customization* for details.

Example Requests for an authorization code flow

The following are sample requests for an authorization code flow, this can get you started, but we recommend learning about Open Id Connect and specifically Auth0 to get the most out of your authentication and authorization.

1. Obtaining the authorization code

http

```
GET https://infosubscription.eu.auth0.com/authorize?response_type=code&client_id=YOUR_
↪CLIENT_ID&connection=CONNECTION&scope=openid&redirect_uri=https://YOUR_APP/callback&
↪state=YOURSTATE HTTP/1.1
Host: infosubscription.eu.auth0.com
Connection: keep-alive
Accept: */*
```

Listing 7: Sample Response

```
HTTP/1.1 302 Found
Location: https://YOUR_APP/callback?code=AUTHORIZATION_CODE&state=YOURSTATE
```

The sample request here uses *connection* as a parameter, please note that this parameter is not required. Refer to the Auth0 documentation for details (see top section).

2. Exchanging the code with a token

curl

http

```
curl --request POST \
    --url 'https://infosubscription.eu.auth0.com/oauth/token' \
    --header 'content-type: application/x-www-form-urlencoded' \
    --data 'grant_type=authorization_code&client_id=YOUR_CLIENT_ID&client_secret=YOUR_
↪CLIENT_SECRET&code=AUTHORIZATION_CODE&redirect_uri=https://YOUR_APP/callback'
```

```
POST https://infosubscription.eu.auth0.com/oauth/token HTTP/1.1
Host: infosubscription.eu.auth0.com
Content-Type: application/x-www-form-urlencoded
Accept: application/json

grant_type=authorization_code&client_id=YOUR_CLIENT_ID&client_secret=YOUR_CLIENT_SECRET&
↪code=AUTHORIZATION_CODE&redirect_uri=https://YOUR_APP/callback
```

Listing 8: Sample Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "access_token":"eyJz93a...k4laUWw",
    "id_token":"eyJ0XAi...4faeEoQ",
    "token_type":"Bearer",
```

(continues on next page)

(continued from previous page)

```
    "expires_in":86400
}
```

Obtaining the Subscriber Id from the token

During the OIDC flow, the client may request an access_token or an id_token.

When users are managed using the INFO-Subscription Sales Poster and self-service client, both of these tokens should have a custom claim for the subscriber id, similar to the listing below:

Listing 9: Sample Id Token content

```
{
    "https://info-subscription.com/subscriberId": "a9c6b736-dac0-4805-93a2-934ce049551d",
    "https://info-subscription.com/tenantId": "abb7c92e-b8f2-4ae5-fef0-08d69bbc8a54",
    "iss": "https://infosubscription.eu.auth0.com/",
    "sub": "google-oauth2|111745085132080132986",
    "aud": "https://api.info-subscription.com/",
    "iat": 1559798200,
    "exp": 1559805400,
    "azp": "zMOPVqHu29qTWkzqJ6Ybh3Eudohz45v8",
    "scope": ""
}
```

This basically identifies the logged in user as a specific subscriber. It is the subscriber that owns a subscription, not a user. In practical terms this means that multiple users can be related to the same subscriber.

Important: Not all users have subscribers!

When you have obtained a *SubscriberId*, head on to the *Authorizing Subscribers* section for details on how to determine if you should let the subscriber access a given resource.

Login Page Customization

The Login page that the uses lands on when starting the Authorization flow currently allows for a few customizations that should be added as extra parameters in the query string.

The parameters that you can set are described in the table below together with the effect it creates in the login page.

Table 7: Login Page customizations

PARAMETER	DESCRIPTION	EFFECT
orgname	The name of the organization as defined in INFO-Subscription	Customized Window Title and Logo on Login Page
language	Auth0 language code (https://github.com/auth0/lock/tree/master/src/i18n)	Displays the login page in the specified language

An example for setting the Organization to *Demo* and the Language to *Swedish*:

```
https://infosubscription.eu.auth0.com/authorize?response_type=code&client_id=YOUR_CLIENT_ID&connection=
/YOUR_APP/callback&state=YOURSTATE&language=sv&orgname=demo
```

Advanced Scenarios

There are several advanced scenarios such as

- Keeping a user signed in using refresh tokens
- Not prompting for login if already logged in elsewhere
- Passwordless signin
- Probably more!

All of these scenarios are described in detail in the Auth0 documentation, so we recommend you head over to https://auth0.com/docs/ for these advanced scenarios.

2.11.2 Authorizing Subscribers

Since the business defines the auhtorization procedure, this documentation will not provide a definitive authorization procedure. This is an outline of an authorization procedure which will fit many businesses, and it assumes the use of the user model provided with INFO-Subscription.

The process is outlined here:

1. Determine which organization(s) the application is authorizing for (typically determined by the site/context)
2. Start an interactive login flow with Auth0 and let the user login.
3. Obtain the *SubscriberId* from the the id token.
4. Get all subscriptions for the obtained *SubscriberId* and *OrganizationId* from the first step
5. If one or more subscriptions covers the current date (between starttime and endtime), the subscriber has access.

Important: Remember to check that the subscription covering the current date is indeed still valid, i.e. when was it cancelled if it was cancelled.

Expanding Authorization

There are a slew of options for authorization, the following outlines some approaches to obtain more information for a subscription which may help in granting or denying access.

Authorizing by plan or products

Each subscription is associated with a plan (package in the API), each plan contains one or more products. It is possible for the application which is authorizing to have a list of Plans or Products which grants access to various items.

Plans are typically used for bundling things together and special offers, and as such products typically have a closer relation with authorization needs, but it depends on the business what is most appropriate.

The basic flow would then be adapted with steps:

1. Let some administrative user map products by presenting them with the products list and keep the mapped list of product ids.

2. For all subscriptions thats cover the current date, verify that one/all products in the defined list is included in the subscription.

Note: In case of multiple subscriptions decide if a multi-product requirement can be met from different subscriptions or not.

Important: A subscription may be cancelled, in which case it will have an indicator of being cancelled and when the cancellation is effective from (*CancellationTime*). Remember to verify these field if access should be removed immediately and not at the end of the original subscription period.

Authorizing by Enterprise Plan

In some cases the application might not be authorizing for product access, but instead a report/dashboard view of an enterprise agreement. For this scenario, adapt the login flow to determine if the Enterprise Plan of the subscription matches with the plan(s) for which information is required.

Authorizing historically

Some businesses may require historic accuracy when authorizing. Typically that would be the case when granting access to an archieve or similar.

Modifying the basic flow as follows:

1. Determine the date or date-range of the archive item.
2. Determine if the subscription should fully cover the range, or just overlap partially
3. When filtering for subscriptions don't use current time but use the time of the archive item.

2.12 Custom Domain Support

By default all tenants utilizing the INFO-Subscription self-service and salesposter solution will be hosted on a generic azure domain name in the form, *https://{tenantName}-s4.azurewebsites.net*.

While functional, it may be desirable to have a custom domain, which is possible but currently requires some manual work by Infosoft support. If a custom domain is required please *contact support*.

2.12.1 Consequences of enabling Custom Domains

Simply put there are no direct consequences of establishing a custom domain.

The old domain will keep on working without issues, and the new one will also be accessible.

All example URLs in the documentation that refers to the self-service site can be replaced with the custom domain name.

2.13 Events and Webhooks

Events describe things that has occured inside INFO-Subscription and can be used by a client to react to these changes.

Webhooks are the mechanism used for notifying clients about the various events and at its simplest it is just a `POST` request with a payload.

When an event is triggered the system will send an event to all registered webhooks with a payload describing the event.

Note: Typically when an endpoint is configured to receive events, the terminology used would be that the endpoint *subscribes* to the event and that configuration would be denoted a *subscription*.

In an attempt to avoid confusion with the subscriptions provided by INFO-Subscription, the documentation will use a slightly different terminology, replacing *subscribe* with **receive**, and *subscription* with **registration** or **webhook**.

2.13.1 Enabling Webhook Event Notifications

Since not everyone needs event notifications the infrastructure required is not enabled by default on a tenant.

Enabling notifications is a one time operation, and there is currently no way to disable it (don't worry its not dangerous :))

There are two ways to enable event notifications:

1. Press a button in the Administrative UI
2. Use the Notifications API with an empty `POST` request

Deploying the infrastructure takes a few minutes, so go ahead and grab a cup of coffee and it will probably be done by the time you get back.

2.13.2 Registering webhooks

A client may receive events by registering one or more endpoints with the tenant. Each endpoint must specify one or more events to be received.

You can create as many registrations as you want, and one event can go to multiple registrations.

It is recommended that you have one registration per endpoint. In case you want multiple events for a given endpoint, just specify multiple events - it is possible to update them later on.

Important: Webhooks requires `HTTPS` endpoints.

Let's Encrypt makes certificate handling easy, so there are no excuses!

Webhook Validation Handshake

Behind the scenes INFO-Subscription use Azure Event Grid for the actual event delivery, which requires a validation procedure. This validation procedure is to make sure you actually own the endpoint you are registering.

If you are familiar with Azure Event Grid and its validations, just do what you normally do, otherwise read on.

At the time of registration as special "validation event" is sent to the endpoint. It looks like any other event, but the content contains a `validationCode` property.

The application should verify that this is indeed something it wants to handle and respond with a simple json object containing the validationCode in the `validationResponse` property.

The following is a complete example of the validation event, note the `eventType` and the content of the `data` property.

Listing 10: Validation Event

```
[{
"id": "2d1781af-3a4c-4d7c-bd0c-e34b19da4e66",
"topic": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
"subject": "",
"data": {
    "validationCode": "512d38b6-c7b8-40c8-89fe-f46f9e9622b6",
    "validationUrl": "https://rp-eastus2.eventgrid.azure.net:553/eventsubscriptions/
↪estest/validate?id=B2E34264-7D71-453A-B5FB-B62D0FDC85EE&t=2018-04-26T20:30:54.4538837Z&
↪apiVersion=2018-05-01-preview&token=1BNqCxBBSSE9OnNSfZM4%2b5H9zDegKMY6uJ%2fO2DFRkwQ%3d"
},
"eventType": "Microsoft.EventGrid.SubscriptionValidationEvent",
"eventTime": "2018-01-25T22:12:19.4556811Z",
"metadataVersion": "1",
"dataVersion": "1"
}]
```

The following is an example of an "approved" response to the above validation event

Listing 11: Validation Response

```
{
    "validationResponse": "512d38b6-c7b8-40c8-89fe-f46f9e9622b6"
}
```

Manual Validation Handshake

If you do not have total control over the webhook, validation using the handshake may be hard or imposible, typically the case when you use a third party integration service such as Zapier or IFTTT, where the response is not synchronous but instead always a 200 OK with the real response being generated later.

For these scenarios you can configure the webhook such that you get the event data, and a `validationUrl` is included as part of the data. Look into the event data log of your platform to find the URL. A manual `GET` request (i.e. copy/paste to your favorite browser) will then approve the endpoint (or you can have your platform visit the URL automatically).

The URL is valid for 5 minutes after registration has started.

It is not possible to do such a manual registrations with the UI, so you need to supply the registration using the register web hook API and supplying the parameter `disableValidation`. If the registration is not approved within 5 minutes it will be rejected.

Direct Event Grid integration

If you are using Azure services or use products with some sort of event grid integration built in, things are a lot easier with direct access to the event grid resources.

While not an out-of-the-box experience, we might be able to work something out for your particular needs, so please contact support and we will have a look.

2.13.3 Events Schema and Payload

All events follow a common schema with some "global" properties.

The following is the json schema for an event

```
[{
    "topic": "string",
    "subject": "string",
    "id": "string",
    "eventType": "string",
    "eventTime": "string",
    "data":{
     "object-unique-to-each-publisher" : "values"
    },
    "dataVersion": "string",
    "metadataVersion": "string"
}]
```

The table is a short description of each property.

Property	Type	Description
topic	string	Full resource path to the event source.
subject	string	Publisher-defined path to the event subject. Subject means what the event concerns not the title.
eventType	string	Defines the type of the data
eventTime	string	The time the event is generated in UTC
id	string	Unique identifier for the event
data	object	Event data specific for the given eventType
dataVersion	string	The schema version of the data object
metadataVersion	string	The schema version of the event metadata. Event metadata being the properties described here

Event Data

Each event contains a property `data` this data property is essentially the event content that is interesting. It is a `JSON` object, and the `eventType` property describes which object instance it is.

To learn more about the available events and what data they contain see the API Reference in the Models section.

Each event has a typename in the following format `com.info-subscription.{eventName}`, so the `SubscriptionCancelled` event would be `com.info-subscription.SubscriptionCancelled`. This is also how the events are documented in the Models section of the API reference docs.

Event Headers

All webhook/event requests will include a series of custom header values that may be used for different purposes.

The current list of headers sent to all endpoints is listed in the following table.

Header Key	Type	Description
S4-TenantId	string/uuidv4	Unique identifier for the tenant.

2.13.4 Events Philosophy

There are a few different approaches to take when designing events. This section attempts to clarify some of our reasoning, you might not agree, but atleast you will know why.

Lightweight Payload

We strive to follow a philosophy of lightweight event payload with only the minimum of information needed to make the event sensible, while also providing key information to query for details.

The reasoning behind this is that event occurrence in itself, together with the subject, is often enough to react to a given event without any futher information. For instance the `SubscriptionCancelled` event contains the `SubscriptionId` and the `SubscriberId`, the first being the subject of the event, i.e. which subscription was cancelled. The later is such a common additional piece of information that describes the owner, so it is included as well.

The fact that a subscription was cancelled, and which one it was, is enough to remove access to a web page, or send an "Sorry to see you go" email, or any other number of actions.

In some cases the receiving application might want more data, in which case the API can be queried using the `SubscriptionId` or the `SubscriberId`.

Keeping the events lightweight, makes it simpler for us to deliver the events as fast as possible.

Registration for specific events

Often a receiving application will only care about a handful of events, and it will have to discard the rest.

Since events which are just discarded is a waste of bandwidth and processing time, each webhook registration must specify which events it wants.

For the few applications that handles many events in a single endpoint, there is a performance consideration to make. Perhaps the endpoint is perfectly capable of handling todays events, but then we add another event to INFO-Subscription and all of a sudden no events are properly handled because the endpoint it overwhelmed. By not allowing for wild-card/all type event registrations, we help alleviate that problem and reduce stress on your application as well as on the INFO-Subscription infrastructure.

By no means are these considerations enough to handle all issues, but it helps - and it is not immensly annoying or troublesome for any of the involved parties.

2.13.5 Event Delivery

Since INFO-Subscription relies on Azure Event Grid for event delivery, the delivery rules/details are defined by Event Grid. The following is a short summary of those details, for the full information please refer to https://docs.microsoft.com/en-us/azure/event-grid/delivery-and-retry

Event Grid provides durable delivery, which basically means it will retry if it did not get a proper response from the endpoint.

Success Codes are defined as

- 200 OK
- 202 Accepted

Failure to respond within 50 seconds will be counted as a delivery failure, and typical HTTP error codes such as 400, 404, 500 and 503 will also mark the delivery as failed.

If a delivery has failed it will be retried according to the following "timetable"

1. 10 seconds
2. 30 seconds
3. 1 minute
4. 5 minutes
5. 10 minutes
6. 30 minutes
7. 1 hour
8. Every hour untill 24 hours

After 24 hours, the event is considered lost and no futher attempts are made.

Security/Authentication

Once a registration has been created, there is currently no built security mechanism such as an API key or HMAC to verify the validity of the content.

You can append a secret as a query parameter when creating the webhook registration which will then be sent with every request if the event originates from INFO-Subscription.

2.13.6 Available Events

At any given time, INFO-Subscription exposes the events listed in the API Reference in the Models section.

The following contains a more detailed description of some of these events, when it is triggered and examples/ideas of what a given event might be used for.

Event: com.info-subscription.SubscriberCreated

Not suprisingly, whenever a new Subscriber is created, it triggers an event `SubscriberCreated`.

There are multiple sources for new Subscribers, but most typically it is done during registration of new Orders.

The event contains the SubscriberId an optional ExternalId and the Subscriber Number (whether its generated or externally defined).

Example Use cases

The related events for Subscriber and Subscriber Contact may typically be used to synchronize with external CRM, Datalakes, Datawarehouses or similar. Mapping and correlation with other sources may be done via the External Id or via the Subscriber Number depending on the design.

If done properly, such a synchronization opens up multiple use cases such as, customer segmentation, marketing/campaign analysis, reporting, visualization, customer communication etc.

Related events:

- `com.info-subscription.SubscriberDeleted`
- `com.info-subscription.SubscriberContactCreated`
- `com.info-subscription.SubscriberContactUpdated`
- `com.info-subscription.SubscriberContactDeleted`

Event: com.info-subscription.OrderProcessed

Once an order is completed and processed an event is triggered called `OrderProcessed`. This indicates that the platform has accepted the order and the Subscription should start as expected.

Orders can be registered via the Salesposter, the Merchant UI or a custom solution using the Orders endpoint in the API.

The event contains a few minimal key points about the order, as well as an indicator about whether the system has finished creating the first subscription period at the time of the event triggering.

Example Use Case

The `OrderProcessed` event may be used to start up dialog with the Subscriber in a Welcome email, or to build up real time statistics of sales.

Other uses cases includes handing off initial distribution information such as a startup one-off delivery, typical for ISPs and Cable operators where some hardware is required to get started.

Event: com.info-subscription.SubscriptionCreated

All operations that results in a new subscription period will trigger a `SubscriptionCreated` event.

The typical sources of a new Subscription are

- Orders
- Subscription Renewal

Other operations may create new subscriptions such as Plan Changes, Pauses and Infosoft initiated migrations (this is not a complete list).

Important: The `SubscriptionCreated` event should not be confused with the `OrderProcessed` event. Subscriptions are created on a recurring basis until a subscription is cancelled, while orders are only created at the beginning of a subscription.

Example Use Case

The `SubscriptionCreated` event may be used to keep external systems up to date instead of doing bulk synchronizations, such as updating:

- External Entitlement/Authorizaitons
- Delivery Management
- Service Desks
- Cash Registers

It may be used to distribute out a monthly newsletter (if the subscription is monthly), or it may be used to notify subscribers that they have a subscription (for infrequent subscriptions).

Event: com.info-subscription.SubscriptionDeactivated

When a subscription has been cancelled, and the cancellation time has been reached/passed, a `SubscriptionDeactivated` event will be fired. For sources of subscription cancellations, please refer to the `SubscriptionCancelled` event below.

Example Use Case

Some ideas on how to use the deactivation event.

- Revoking authorizations
- Starting data cleanup operations, like removing from mailing lists

Event: com.info-subscription.SubscriptionCancelled

Operations that result in subscriptions being cancelled will lead to a `SubscriptionCancelled` event being fired.

Typical source of Subscription cancellations are

- User initiated cancellations (Merchant or Self-Service).
- Orders with an Automatic Cancellation marker.
- Subscription Pauses (which are implemented as Cancel followed by Create).
- Forced cancellations due to lack of payments (Payment Stop).

The `SubscriptionCancelled` event is fired at the registration time, NOT at the time of the cancellation taking effect.

Example Use Case

Similar to `SubscriptionCreated`, the `SubscriptionCancelled` event may be used to keep external system in sync.

- Automated 'Sorry to see you go' emails.
- Automated offers to save the customer.
- Automated transfer to external debt collection (Payment Stops).

Event: com.info-subscription.InvoiceIssued

Whenever a new Invoice is issued/finalized an `InvoiceIsued` event is triggered.

The event is typically triggered during automatic billing routines. It contains a few default properties such as the `DueDate` and the `Number` that it has been assigned.

Other operations that may trigger Invoices to be generated are

- Merchant initiated credits with associated corrections (generates a new invoice).
- Removal of future cancellations (re-instates previously creditted invoices).
- Manual issuance of invoices (rarely)

Example Use Case

The `InvoiceIssued` event may be used to keep external system in sync, typically finance or ERP systems.

The use cases for the event are varied and it can also be used for other things such as:

- Generating an advance notice about an upcoming Invoice.
- Used as a source for distribution/inventory management to pickup new parcels for delivery (with the Invoice).
- External distribution of Invoices, for instance via specialized vendors with support for multiple communication formats.
- As a source document for factoring integrations.

Event: com.info-subscription.ReminderIssued

Reminders are triggered at scheduled interval relative to the original `InvoiceIssued` event.

They are only triggered if reminders are configured, an no payments exist.

Example Use Case

The `ReminderIssued` event is typically used in much the same way as `InvoiceIssued`, such as:

- Generating SMS message about a lack of payment.
- As a source for debt collection integrations.

Event: com.info-subscription.CreditNoteIssued

All operations that result in a new credit note will result in a `CreditNoteIssued` event being triggered.

The event points to the original invoice as well as a few key pointers about the new credit note.

Typical source of the event are:

- Subscription cancellations (future Invoices are automatically credited).
- Merchant initiated credits.

2.14 Subscribers

In INFO-Subscription subscribers serves as the entity which own subscriptions. They are abstract entities, where the only purpose is a unified identification mechanism. A subscriber may be a person, a company, a government, a computer, a pidgeon or whatever else you can imagine, as long as you can imagine that it can subscribe to one of the configured subscription plans.

2.14.1 Subscriber Identity (Identifiers)

Subscribers have two identifiers, one internal UUID/GUID and one external human-readable running number called the subscriber number.

Optionally a subscriber may have an external Id, this is useful to map between INFO-Subscription and external systems.

Unlike a subscription, a subscriber is never associated with an *organization* and is considered global. Keep this in mind if you have to decide on having one or multiple tenants.

2.14.2 Subscriber Contacts

While subscribers may be anything, contacts are considered entities with a name and an address. Typically a private customer would have a single primary contact, which is him/herself. A company will have one or more contacts responsible for managing and/or paying for subscriptions.

The contact information is used when issuing Invoices, and in some cases as the basis of delivery information for distribution of physical goods.

2.14.3 Customizable Subscriber Number

By default the subscriber number is generated from an internal sequence provided by INFO-Subscription.

In some cases it may be desirable to use externally assigned numbers, typically the case where external CRM/ERP systems have numeric identifiers, or multiple subscription systems are in use.

Internal subscriber number generation can be toggled off, see to the refence docs for details.

Note: Toggling off the number generation requires that an external system provides the subscriber number, as this is a required field.

Warning: Switching back and forth between internal and external subscriber number configuration may lead to weird issues as the internal number generator only knows the numbers it has generated.

2.15 Plans and Template Plans

An important concept for all subscriptions are the notion of a plan. At its core, the plan is the content of the contract between the subscriber and the organization, while the subscription is the timeline. The plan encapsulates things such as the agreed price, the products that are included, and the billing frequency.

Note: For historical reasons there are some terminology confusion regarding plans. The API use the terms `package` and `template package` to denote plans and template plans. All places where the API mentions a package, it is in fact a reference to a plan.

2.15.1 Templates, Choices and Subscription Plans (Instances)

All subscriptions needs to be assigned a plan that determines its contract/rules. To help alleviate some of the construction pains of building plans, INFO-Subscription has a concept of template plans

As the name suggests, the template plan is a template with a series of predefined settings that a subscription will use for its plan. When placing a new *order* a template is referred for the various defaults, and possibly some choices that override those defaults or specify a selection.

The template and the choices together form a plan instance, a Subscription Plan, which is associated with the created subscription. When created the subscription plan does not refer back to the template and the template can be changed at will without affecting the instance on the subscription.

For more details on choices during orders refer to the *order section*.

2.15.2 Subscription Plan Changes (Upgrades and Downgrades)

Most subscription business models contains the option to switch between plans, either as an upgrade (typically a more costly product), or as a downgrade (a less costly product).

INFO-Subscription supports upgrades and downgrades through Subscription Plan changes.

Note: There is no logical concept of an upgrade or a downgrade, the definition of whether a change is an upgrade or a downgrade is left to the client.

Doing a plan change is simple as:

1. Determine the *SubscriptionId* to change.
2. Determine/Build the SubscriptionPlan to change into (refer to Orders for details on Plan Choices)
3. Schedule the change

A change is defined with a pocessing type, denoting when the change should be carried out.

1. Immediately - Process now
2. OnScheduledTime - Process at the defined time
3. OnRenewal - Process when the subscription is renewed next

Immediate Changes

Immediate changes are carried out at once, with no options for regret other than somehow creating compensating transactions.

Essentially an Immediate change means the current subscription is cancelled now, and a new one is generated with the new parameters. Outstanding allowances or charges are transferred to the new subscription and the current time period is prorated according to the billing configuration.

Changes On Scheduled Time

Scheduled changes are carried out around the time supplied with the change request. Very similar to Immediate changes.

Any proration happens at the time of the execution of the schedule, and not at the time of registration.

Mostly useful for scenarios where the is a fixed delay related to the desired change. That might be a planned summertime product downgrade for instance.

Scheduled changes can be revoked by deleting them before they are executed.

On Renewal Changes

Changes scheduled for the renewal are a bit different from the other two types.

When a subscription is billed, changes registered for renewal are planned into the billing flow entirely, and the running subscription is never cancelled, nor does any proration occur.

Changes are effective on the next subscription and automatically associated with the renewal process.

This is useful for business models operating with entire periods for downgrades and other similar non-cost related changes.

2.16 Creating new subscriptions using orders

An order is an encapsulation of a series of steps/stages to help create new subscriptions and keeping the flow in a semi-atomic transaction, meaning that if an order fails to complete (or is never completed by the user) there is no clean-up duty for the client, and when an order completes there should be no series of followup steps such as making sure the first invoice is generated etc.

While it is possible to create subscriptions without using an order, we recommend that clients use orders to simplify the process.

An order is split into several steps, primarily to ensure consistency between the various phases and transactions that an order covers.

1. Order Initialization - Captures and calculate information such as the selected subscription plan, price and which payment agreement to use
2. Payment Agreement Registration - Typically defers to third parties for payment processing. Only required for certain electronic payment providers (Creditcards and Mobile Payments typically)
3. Order Completion - Gathers information from the previous steps and finalizes/activates the subscription.

Applications/clients might want to prepare orders, for instance to send out pre-filled order details for acceptance by customers or other similar scenarios.

Note: There might be timeout/expiration issues with this approach if using online Payment Service Prodivers (PSPs) such as Swedbank Pay. because the order initialises the payment (at the time of writing the session is valid for two weeks using Swedbank Pay for instance)

2.16.1 The anatomy of an Order

The main thing to remember about orders, is that they are not actually resources/entities as such, but rather encapsulates parameters of a workflow. That means that an order, once completed is of little/no future use except perhaps for reporting and statistics purposes.

An order contains information about a few things

- Who/What is subscribing to an offering - the `subscriberId`
- What offering are they subscribing to - the `templatePackageId` (see *plans*)
- How should payment be processed, specified by either of the following:
  - The `paymentAgreementId`
  - The `paymentAgreementParameters`
- Who is offering this subscription - the `organizationId`

The are other fields, all of which come into play in various scenarios.

Package Choices/Overrides

As the name implies a `TemplatePackage` is a template for the package/plan that will be created for a given subscription.

Some of the parameters of a package may require choices by the subscriber (or the sales process), these choices are often optional, but can be defined using the `templatePackageChoices`.

To figure out which things are overridable by the user, have a look at the specific template plan as it defines the flexibility.

The choices are validated when creating the order, and again when completing it, so it should not be possible to define an illegal order if for some reason the template is changed.

Note: There is some confusion about the terminology related to packages and plans. The word Package is used in the API while the term Plan is used for the self-service and Merchant UI. They are in general synonymous.

Processing Payments

Most of the time, there is a need to enact some sort of payment process.

When an order is created, if the paymentMethod is set to `PayEx` or similar, the response will contain a `terminalRedirectUrl`.

This URL (Uniform Resource Locator) indicates where the client should send the user so he/she can input the payment card details.

The parameters must contain a return url and a cancel url (they can be the same). When the user completes or cancels the payment process, he/she is returned to either of these pages, with an Id in the querystring representing the order.

Depending on where the user was redirected the client can do whatever is needed to update local systems and either *Cancel* or *Complete* the order.

Invoice Contact/Address

INFO-Subscription manages invoice addresses by creating a separate subscriber contact (see: *Subscribers Section*) and associating it with the subscription.

It is possible to either define an existing contact id in the *invoiceContactId* or by defining details for a new contact (to be associated with the ordering subscriber). The details to provide are the same as for a generic subscriber contact.

2.16.2 Orders and Subscribers

Subscribers can either be created before an order, or during the order flow, if the client creates subscribers the *subscriberId* must be specified with the order.

In case the subscriber should be created with the flow, an *externalSubscriberId* and/or *subscriberNumber* must be provided.

2.16.3 Completing an Order

Completing an order takes a bit of time, so we suggest the client presents the user with some sort of processing feedback.

Completing the order currently executes a few different tasks such as

1. Builds a custom plan for the subscription and verifies the result is valid.
2. Complete the transaction at the PSP to ensure that the agreement can be used (if applicable).
3. Creates a PaymentAgreement for the given Provider.
4. Creates a subscription with the defined PaymentAgreement as the payment method.
5. Schedule a payment demand with the amount from the order and a due date which is the same as the subscription start.

Following these steps another series of steps will be enacted by the billing engine.

1. Create and Issue an Invoice for the Payment Demand (based upon the schedule this may be immediately or in the future).
2. On the due date, initialize a payment request of the demand.
3. Creates a payment representing the captured amount.
4. Create a draft demand and invoice for the next period.
5. Schedule a payment demand for the next period.

These additional things happens asynchronous, so don't expect it all to be completed the second the order response is generated. But it basically means you can direct the user to a payment/invoice overview and within a short time they will see their invoice and/or payment.

Note: The steps carried out by the billing engine after an order has completed are nearly identical to the steps carried out during recurring billing processing.

Once completed the response contains an updated order view with the new status and various Id's that informs the client of what was created.

While not immensely useful the order will persist so the client could choose to have a list of *purchases/orders* or similar to show historic orders.

Subscriptions generated by the passing of time (i.e. recurring subscriptions) are not treated as orders and won't mess up the view.

2.16.4 Cancelling an Order

If for some reason the user opts to cancel the payment process or the order, we recommend that the client explicitly cancels the order.

While not strictly necessary it helps with a few things

- The order is set as cancelled and no further attempts to process it can occur.
- If applicable, any payment process at the PSP is cancelled.
- Statistics will be more accurate.

The fact that an order was cancelled might be useful to business people to follow up during various marketing/sales campaigns or similar activities.

Automatic cancellations and completions

In some cases the INFO-Subscription will attempt to observe the order process and automatically close any open orders.

At the current time this is only appropriate for orders with *Vipps* as a payment agreement.

If an order is not completed, the upstream Vipps agreement is queried for its state after some time has passed (roughly 5 minutes) If the Vipps agreement has been approved, the order is automatically completed. If the Vipps agreement has been rejected or expired, the order is automatically cancelled.

Quite a few consumers close the shopping/browser window after being redirected to the Vipps App, and thus never completes the order successfully. Similarly if the Vipps agreement has expired, there is no way the order can ever be successfully completed, so it might as well be cancelled.

2.16.5 PSP Callbacks

Many PSPs have a concept of a *callback* used for ensuring that payments are processed correctly in the event of a client failure. Typically failures are things such as loosing internet connectivity, user closes browser session, browser/machine crashes, appliction errors and the list goes on.

The idea is that the PSP will do a *callback* to a registered URL out-of-band from the browser.

We recommend that the client implements some sort of callback handling that will either *Cancel* or *Complete* the order.

Important: INFO-Subscription currently has no built in callback handling that can be utilized but it is on the roadmap. The facility of a callback would be similar to that of automatically cancelling or completing orders.

2.17 Order Examples

The following contains a few common scenarios/examples for the ordering process. Unless otherwise stated, each of them presents the body for creating a new order using an order request like the following.

Listing 12: Create a new order

```
POST /Order/ HTTP/1.1
Host: https://api.info-subscription.com
Content-Type: application/json
S4-TenantId: SOMETENTANT
Authorization: Bearer RANDOMTOKEN
Content-Length: 10
```

Each example typically only shows properties relevant to the example and omits required elements for brevity.

It is possible to combine many of the examples, though not necessarily all of them.

2.17.1 Example: Choosing Products On A Plan

For organizations with many products to sell, it is often required to have the option to construct any product combination the subscriber desires. Consider an traditional cable TV provider, many of them offer extra channels, and even serves as an Internet Service Provider.

Instead of creating every combination of product selects as template plans, it is possible to provide one or two template plans that allows choosing between products. Continuing the example of a TV/ISP company, they may have the following products

- Basic TV `00000000-0000-0000-0000-000000000001`
- Premium TV `00000000-0000-0000-0000-000000000002`
- Sports Channel `00000000-0000-0000-0000-000000000003`
- Movie Channel `00000000-0000-0000-0000-000000000004`
- Random Channel `00000000-0000-0000-0000-000000000005`
- Slow Internet `00000000-0000-0000-0000-000000000006`
- Fast Internet `00000000-0000-0000-0000-000000000007`
- Ultra Internet `00000000-0000-0000-0000-000000000008`

An example template would look something like the following:

Listing 13: Template Plan/Package Headers

```
POST /Order/ HTTP/1.1
Host: https://api.info-subscription.com
Content-Type: application/json
S4-TenantId: SOMETENTANT
Authorization: Bearer RANDOMTOKEN
```

Listing 14: Template Plan/Package Body

```
{
    "id": "10000000-0000-0000-0000-000000000000",
    "products": [
        {
            "productId": "00000000-0000-0000-0000-000000000001",
            "fullPrice": 1000,
```

(continues on next page)

(continued from previous page)

```
            "taxPercent": 25
        },
        {
            "productId": "00000000-0000-0000-0000-000000000002",
            "fullPrice": 2000,
            "taxPercent": 25
        },
        /* Omitted for brevity  */
        {
            "productId": "00000000-0000-0000-0000-000000000008",
            "fullPrice": 8000,
            "taxPercent": 25
        }
    ],
    "name": "TV and Internet",
    "description": "Internet and TV For All Yo!"
    /* Rest omitted for brevity */
}
```

If one were to subscribe to the above *TV and Internet* package, with no choices, it would contain 8 Products.

However, using choices, it is possible to construct and order for *Basic TV* and *Slow Internet* like the following

Listing 15: Order of Basic TV and Slow Internet using choices

```
{
    "subscriberId": "11111111-2222-3333-4444-000000000000",
    "templatePackageId": "10000000-0000-0000-0000-000000000000",
    "paymentAgreementId" : "44444444-5555-6666-7777-000000000000",
    "organizationId" : "20000000-0000-0000-0000-000000000000",
    "templatePackageChoices":
    {
        "products": ["00000000-0000-0000-0000-000000000001", "00000000-0000-0000-0000-
↪000000000006"],
    }
}
```

2.17.2 Example: Overriding Pre-Defined Prices

Typically a template plan is setup with a specific set of products.

The price is either defined directly on the template plan, or as a function of the list price on the products.

In some scenarios it is desirable to override this price when creating the order.

For instance where an extra-ordinary discount should be given to a subscriber (for whatever reason), or in cases where each subscription is negotiated by a salesperson, which is typically for high-value enterprise and B2B susbscriptions.

It is possible to override the price during the order like the following

Important: It is important to note that this is only possible if the template is configured to allow price overrides!

2.18 Payment Agreements

INFO-Subscription attempts to manage all parts of the billing lifecycle in an automated fashion. When a payment demand is generated, and an invoice issued, INFO-Subscription refers to the current payment agreement, in order to determine how to get money from the subscriber.

A payment agreement is a reference between the subscriber and the tenant which allows the tenant to claim payments directly from the subscriber.

Different agreement types provides different capabilities and routines.

All subscribers can have a payment agreement called *None* or *InvoiceOnly*. This basically means there is no way to claim the amount so the system does no attempts at generating a payment.

2.18.1 Properties of a Payment Agreement

All payment agreements have a few important properties

- A *PaymentType*: What sort of payment mechanism is this: DirectDebit, Mobile App, Payment Card (VISA/Mastercard etc.), eInvoice.
- A *PaymentProviderType*: Which provider is used for the underlying payment mechanism.
- A reference for more provider specific details.

The *PaymentType* is mostly a cosmetic/informational property, especially for tenants with only a few integrations or operating countries.

The *Payment Provider Type* indicates the service provider that INFO-Subscription uses to facilitate the given claims. At the time of writing the following providers are currently supported:

- AvtaleGiro, DirectDebit by Mastercard Payment Services.
- Card Payments, by Swedbank Pay (formerly PayEx eCommerce).
- Vipps Recurring, Mobile Payments by Vipps.
- eFaktura, eInvoicing by Mastercard Payment Services.
- Autogiro, DirectDebit by Bankgirot.
- EHF, Norwegian version of EUs PEPPOL BIS Billing Standard.
- Email, As the name says it handles Email. Can be used to filter Invoice for e-mails but there is no built in Email sending.
- InvoiceOnly/None - the default payment type that does “nothing”.

2.18.2 Managing Payment Agreements

There are multiple ways to create a payment agreement for a subscriber.

- Created during the order registration.
- Created as a self-service action (initiated via a website or target email campaigns etc).
- Initiated out-of-band.

All of the approaches share the same organization/abstraction model in INFO-Subscription and follow the same flow in terms of the integration/API.

1. A Provider Agreement is created (the details vary by provider).

2. A Payment Agreement is created pointing to the provider agreement.
3. The Payment Agreement is registered for a subscription (sometimes this happens automatically).

The idea here is that each Payment Provider, such as Vipps, SwedbankPay or AvtaleGiro, have their own percular details on how to register an agreement, and they have different terminology and different properties availble on the agreements. To abstract away some of these details most of the time, the subscription points to a payment agreement, which in turn points to the provider.

This allows most integrations to just query for the Subscription and Payment Agreement information without worrying too much about the various providers. When a new provider is added, no adjustments are needed to provide general information about these new agreements, only in-depth information requires additional integration.

For details on how to register new provider agreements, refer to the individual provider sections.

A request to create a new payment agreement might look like:

```
{
    "subscriberId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "providerType": "AvtaleGiro",
    "providerAgreementReference": "00000000000011",
    "paymentMethod": "DirectDebit",
    "providerAgreementId": "945a996e-ac8b-4a96-96c3-deb136dc2830"
}
```

Switching Agreements Manually

Once a Payment Agreement has been created, switching to this agreement is a simple as assigning it to the subscription.

Note: Agreement management is also built into the Self-Service client and to some degree into the Merchant client.

Out Of Band Agreements and Automatic Agreement Registration

Some payment providers are typically registered in a process not connected to the merchant, and by extension not in their systems. At the current time of writing, this is relevant for the following providers:

- AvtaleGiro
- eFaktura (Norway)
- Autogiro

For all of the above, the registration will happen, or may happen, in the subscribers online banking solution.

There are different details to be aware of, but in general terms the following process is enacted.

1. The Subscriber registers an agrement in his/her bank.
2. A notification is generated to the merchant.
3. INFO-Subscription registers a *ProviderAgreement* and a *PaymentAgreement*.
4. Once the *PaymentAgreement* is generated INFO-Subscription will attempt to automatically switch to the created agreement.

At the current time, only subscribers on InvoiceOnly/None agreements are automatically switched. It is assumed that whatever process they entered into is preferable from their point of view untill otherwise dictated. At the current time there is no way for a merchant to define a preferred order of payment methods.

Importing Agreements

Sometimes it is necessary to import payment agreements from external sources.

There are several use cases for importing agreements, but the two main scenarios are:

1. When migrating existing agreements from another billing/subscription system.
2. Integrating with ecommerce platforms that generates the agreemt during the initial sale.

Each provider integration exposes its own endpoint for importing existing agreements. There are some variations, so look to the provider descriptions for the details.

Common for all the scenarios is that once an agreement has been imported for the provider, it should be added as a general agreement, and handled the same way as with new orders or during self-service registrations, as *described above*.

2.19 Provider Integrations

2.19.1 AvtaleGiro

AvtaleGiro is the primary (only) Norwegian Direct Debit solution for consumers, it is provided by the Norwegian banks and driven by Mastercard Payment Services (MPS), formerly NETS.

Mandate Registration

In AvtaleGiro terminology an agreement is called a mandate. The following will use the term agreement and mandate interchangeably.

AvtaleGiro traditionally provides an "out-of-band" style agreement registration.

1. The customer receives a physical invoice with a customer identifier (KID).
2. The customer registers this in the online banking solution (or some other means as described by MPS https://www.mastercardpaymentservices.com/norway/utvikler/paymentmandates)
3. A mandate is sent to the merchant.
4. Recurring payments can be charged on the consumers account, associated with the specific mandate.

Since all of this is "out-of-band", there is little developer integration here.

The requirements is that an account is setup for AvtaleGiro (Account endpoint) and that mandates are imported using the OCR Payment file from MasterCard payment services. INFO-Subscription takes care of the rest.

Manually Registering Mandates

A mandate registration requires the following

- A KID (an identifier)
- A SubscriberId
- An AccountId

The mandate is created by sending a POST request to the mandate endpoint, an example request body might look like:

```
{
    "accountId" : "a28373e1-5733-4682-98f0-549cb59800f8",
    "subscriberId" : "acecdb1d-b78b-45bd-9374-4feee7edaf72",
    "kid" : "00000000000011",
    "notificationDisabled" : true
}
```

The output of such a request is a new Id for the mandate, which can then be used to register a general payment agreement as described in the section for *managing agreements*

Connected/Interactive Mandate Registration

Mastercard Paymentservices provides an alternative solution to the out of band registration option, the branded name for this is Mastercard Payment Services' E-Agreement service. Basically its a website that allows for the registration to happen in a connected/interactive manner without the need for the consumer to go to the online banking solution.

The process for such a registration is

1. Create/Get the Subscriber. The Subscriber Number is required.
2. Get the AvtaleGiro bank account number using the account endpoint Account endpoint.
3. Calculate the KID using the External Identifier calculation endpoint.
4. POST the required information to the eAgreement service (refer to MPS Develop docs for the details - https://www.mastercardpaymentservices.com/norway/utvikler/pvugetstarted).
5. If an OK response is received from MPS the Mandate can be imported with the defined KID.
6. Create a general payment agreement with the mandate.
7. *(Optional)* Order a subscription with the agreement created in the previous step.

Note: This is NOT built into INFO-Subscription at the time of writing, but we are looking into adding it as an option to our self-service and ordering process. This will need to be built by all parties not relying on the INFO-Subscription ordering process and self-service solutions regardless.

Cancelling Mandates

At any point in time, the consumer may cancel the mandate, in which case an alternate means should be used to get the payment.

INFO-Subscription manages this scenario by:

- Disabling the mandate once a cancellation is received.
- Disabling the payment agreement and reverting to the default agreement (Invoice).
- Generating reminders on Invoice since no payment are going to be received with AvtaleGiro.

It is possible to cancel a mandate using the mandate endpoint.

At the time of writing, there are no common scenarios where this is needed, but the most likely would be some external lifecycle management of the mandates.

Creating Claims

A Claim is the AvtaleGiro terminology for doing a debit/charge on the consumers account. Once submitted, if not cancelled by the merchant, will lead to an account transfer on the given due date.

Claims are automatically created and cancelled for subscriptions on an AvtaleGiro mandate. There should be minimal need for manually creating claims. Please let us know if you have specific scenarios that is not supported.

Claims can be created directly using the API if required, using the claims endpoint.

> **Caution:** All mandates are created with a maximum debit amount per month. Using the subscription mandate for external charges runs the risk of exceeding this amount. This in turn leads to rejected claims and unpaid invoices.
>
> Use this feature with some caution.

2.19.2 eFaktura

eFaktura is a Norwegian eInvoicing solution providing consumers with an electronic invoice directly within their online banking system. It is provided by Master Card Payment Services.

Unlike many of the other recurring payment systems, there is no automatic payment, the consumer still has to accept the Invoice, but doing so is almost trivial.

Consigment or Mandate Registration

The terminology has historically been a bit confusing, but similar to AvtaleGiro a organization needs a consignment or agreement or mandate from the Subscriber to send him/her an efaktura Invoice. efaktura registration is an "out-of-band" process where the subscriber accepts eFaktura in general (not for the specific organization). Then it is up to the Organization to determine if the subscriber should have eFaktura or not.

INFO-Subscription implements eFaktura with the following process

1. An eFaktura Search is done on a regular basis.
2. New agreements are created (if not blocked).
3. Cancelled/Blocked agreements are removed.
4. Running subscriptions are automatically switched to new eFaktura agreements.
5. Recurring payments are invoiced using the new agreements.

Since all of this is "out-of-band", there is little developer integration here.

The requirements is that an account is setup for eFaktura, INFO-Subscription automatically takes care of the rest.

Combining with AvtaleGiro

Technically it is possible to combine eFaktura and AvtaleGiro into a common request, one that provides a visual representation of the Invoice (eFaktura), and one that takes care of the automated payment (AvtaleGiro). At the current time of writing this is not provided by INFO-Subscription.

Cancelling Agreements

At any point in time, the consumer may block the issuer thus cancelling the agreement, in which case an alternate means should be used to get the payment.

INFO-Subscription manages this scenario by:

- Disabling the agreement once a cancellation is discovered (regular scanning)
- Disabling the payment agreement and reverting to the default agreement (Invoice).
- Generating reminders on Invoice since no payment are going to be received with eFaktura.

2.19.3 Vipps Recurring

Vipps Recurring allows organization operating in Norway to offer mobile/app based recurring payment, for subscribers with a Vipps account. It is backed by a credit card, debit card or bank account managed by Vipps.

Requirements for Vipps Recurring

The requirements is that an account is setup for Vipps (Account endpoint). Infosoft is a Vipps partner and we can facilitate the Vipps onboarding process if required, please *contact support* for details.

Agreement Registration

In Vipps terminology an agreement is almost like a subscription. It is an agreement between the Merchant (a INFO-Subscription tenant) and the subscriber to pay for a given product, at a recurring interval. If the subscription plan changes, the agreement should change with it.

Vipps requires an integrated interactive process for registering new agreements. Typically in conjunction with an order.

1. The potential subscriber selects a Subscription Plan to purchase.
2. A draft agreement is generated on Vipps using the subscribers mobile number.
3. The subscriber is redirected to a website to confirm the information (Managed by Vipps).
4. The subscriber confirms the purchase in the Vipps Mobile App (Managed by Vipps).
5. The order and agreement is finalized and the initial payment associated with the order.
6. Recurring payments can be charged on the subscribers account/card, associated with the specific Vipps Agreement.

Since all of this is an interactive process, some integration is required for it to be available in the order process. INFO-Subscription provides a turn key sales and ordering process that is capable of handling Vipps Agreements.

If the caller utilizes the ordering API provided by INFO-Subscription some of the above is handled, but the initial parameters and the redirection needs to be managed by the client.

In-Shop/Non-Browser Agreement Registration (Merchant Initiated)

Vipps allows a special type of agreement registration where everything is managed by the merchant, and the only thing the subscriber has to do is accept the agreement in the mobile app. This process is almost identical to the above, except the browser redirect part can be headless and without user interaction.

The purpose of this registration process is to provide "in-shop" purchases of subscriptions without requiring the buyer to interact with a browser.

This alternate registration process requires:

1. A special agreement with Vipps that allows merchant initiated agreements.
2. An extra parameter to the order process that indicates the process should be used with Vipps.

Payment Type Management

Since Vipps Recurring is backed by bank accounts and potentially card agreements, there is a need to manage the card agreements upon expiration. All of this is managed by Vipps, refer to the official Vipps documentation for what this entails.

Manually Registering Agreements

It is possible to register/import agreements registered outside of INFO-Subscription.

An agreement registration requires the following

- A Vipps Agreement Id
- A SubscriberId
- An AccountId

The agreement is created by sending a POST request to the agreement endpoint, an example request body might look like:

```
{
    "accountId" : "a28373e1-5733-4682-98f0-549cb59800f8",
    "subscriberId" : "acecdb1d-b78b-45bd-9374-4feee7edaf72",
    "vippsAgreementId" : "agr_123456",
}
```

The output of such a request is a new Id for the agreement, which can then be used to register a general payment agreement as described in the section for *managing agreements*

Creating Charges

A charge is the Vipps Recurring terminology for doing a debit/charge transaction on the subscribers account. Once submitted, if not cancelled by the merchant, will lead to an account transfer on the given due date.

Charges are automatically created and cancelled for subscriptions on a Vipps Recurring agreement. There should be minimal need for manually creating charges. Please let us know if you have specific scenarios that is not supported.

Charges can be created directly using the API if required, using the charges endpoint.

Caution: Depending on the registration process, new agreements may be created with a maximum debit amount per month. Using the subscription agreement for external charges runs the risk of exceeding this amount. This in turn leads to rejected charges and unpaid invoices for the regular subscription.

Use this feature with some caution if the agreement is used by a subscription.

2.19.4 Swedbank Pay

Swedbank Pay is a Payment Service Provider (PSP), formerly PayEx. Swedbank Pay provides a series of different payment services, among these are handling of credit/debit card payments.

Note: Throughout the API and the documentation/description of it, the integration is denoted as "PayEx". PayEx was acquired by Swedbank and the solution was re-branded. In order to not break the API we have kept the naming as-is.

Requirements for using Swedbank Pay

In order to use Swedbank Pay as a card provider, at least one Swedbank Pay account must be configured in INFO-Subscription. This can be acquired either by contacting Swedbank directly, alternatively Infosoft can facilitate the onboarding process if required. Please *contact support* for details.

Agreement Registration

In Swedbank terminology an agreement is basically the same as an agreement in INFO-Subscription. It is an agreement between the Merchant (a INFO-Subscription tenant) and the subscriber, that allows the Merchant to make payment transactions on the subscribers payment card.

Like most other PSP, Swedbank Pay requires an integrated interactive process for registering new agreements. Typically in conjunction with an order.

1. The potential subscriber selects a Subscription Plan to purchase.
2. A payment transaction is started at Swedbank.
3. The subscriber is redirected to a website to card details (Managed by Swedbank Pay).
4. The order and agreement is finalized and the initial payment associated with the order.
5. Recurring payments can be charged on the subscribers card.

Since all of this is an interactive process, some integration is required for it to be available in the order process. INFO-Subscription provides a turn key sales and ordering process that is capable of handling Swedbank Pay Agreements.

If the caller utilizes the ordering API provided by INFO-Subscription some of the above is handled, but the initial parameters and the redirection needs to be managed by the client.

Agreement Registration Without Orders

For existing subscribers on a different payment agreement, or for switching to a newer payment card. It is possible to register agreements WITHOUT an order.

The process is very similar to registrations with an order:

1. The potential subscriber chooses to create a new agreement.
2. A payment transaction is started at Swedbank, with a special indicator that its for agreement creation purposes.
3. The subscriber is redirected to a website to card details (Managed by Swedbank Pay).
4. The agreement is finalized and the initial payment associated with the order.

Prepared Transactions

It is possible to prepare agreement transactions and send them to potential subscribers as direct URLs to the agreement terminal. For more info check the Swedbank website, or *contact support*.

Manually Registering Agreements

It is possible to register/import agreements registered outside of INFO-Subscription. This is typically useful when importing accounts from other sources such as an e-commerce solution.

An agreement registration requires the following

- A Swedbank Pay Recurrence Token
- An AccountId

The agreement is created by sending a POST request to the agreement endpoint, an example request body might look like:

```
{
    "accountId" : "a28373e1-5733-4682-98f0-549cb59800f8",
    "recurrenceToken" : "acecdb1d-b78b-45bd-9374-4feee7edaf72",
}
```

The output of such a request is a new Id for the agreement, which can then be used to register a general payment agreement as described in the section for *managing agreements*.

INFO-Subscription will only accept the agreement if the given *recurrenceToken* is valid on the Swedbank Pay account. Details about the card such as the *expiration* and the *cardMask* may be given during import, but they will be pulled from Swedbank directly if missing.

Creating Payments

A payment or transaction is the Swedbank Pay terminology for doing a debit/charge on the subscribers payment card. Once submitted, if not cancelled by the merchant, will lead to an account transfer on the given due date.

Payments are automatically created and cancelled for subscriptions on a Swedbank Pay agreement. There should be minimal need for manually creating payments. Please let us know if you have specific scenarios that is not supported.

Payments can be created directly using the API if required, using the transaction endpoint.

2.20 Payment Requests

Here is placeholder text

2.21 Reporting and Analytics

INFO-Subscription provides and out-of-the-box database which can be used for reporting, visualization and analytics of your subscription data and economy.

The reporting and analytics database is currently available through direct SQL and indirectly via our hosted Power BI solution.

Many of the data points and visuals available in the Merchant application, as well as all the reports, are based on the data exposed by this reporting database.

Built for reporting, the solution is optimized for read-access and server-based aggregations. Use it to answer quantitative questions about the past or present state of your subscription business.

For details on what type of data is available refer to the datamodel section or explore the model with your favorite MS SQL capable tool.

2.21.1 General Concepts

There are a few concepts that might be relevant to understand about the reporting system.

1. Near real-time

The model is updated in near realtime, mostly depending on the available resources. It is NOT offloaded once a day or once a week or something like that. Data is processed as it is generated in production and reflect into tables based on the stream of events generated by the production platform.

For many reporting and analytics applications this is not terribly important, a delay of 1 day is not a problem. However for it allows for building "read-only" applications on top of the data layer with near-instant access to the production data, in fact a few of the lookup/search features of the Merchant application is built like this.

2. Semi denormalized

It is common knowledge that to be efficient with storage, databases should be normalized.

However when it comes to reporting and analytics, there are a few issues with that approach, the main one being needless joins between tables where you need the data again and again.

To alleviate that issue, some of the tables are built so that common information is denormalized. The two most prominent examples are:

1. Subscribers
2. Organizations

In many places a reference to a *SubscriberId* is replaced with the *SubscriberId*, the *SubscriberNumber* and the primary contact details. Similarly the *OrganizationId* is replaced with the *OrganizationName*, *Address* etc.

The reasoning being that if you want to build a report of all payments for reconciliation or auditing, it is very likely you want to know the subscriber as well. Having it pre-built makes for faster read access at the cost of a bit of storage.

All things are not equal however, so not everything is denormalized. Reach out to *support* if you think something should be denormalized.

3. Pre-Aggregated KPIs

Some KPIs are so common, that it would be a waste of time for everyone to retrieve thousands and millions of rows just to summarize them.

INFO-Subscription provides its own set of KPIs that are aggregated during generation (in realtime).

The enables faster generation of visuals in reports and dashboards.

As with the normalized data, there might be something that you think we should include, then feel free to reach out to *support*, we may be able to acommodate you.

4. Area or Silos of data

The reporting database is divided into areas or silos with different data being put into different areas.

The silos exists as a grouping mechanism, and some data may be duplicated between silos. Mostly they are there to remind consumers that not all concepts map 1:1 and that items you think relate to a specific thing may actually have multiple sources or be connected in a different way than what you expect.

Keep that in mind when you correlate data between silos.

2.21.2 Connecting

At the current time of writing, direct SQL based access is granted to the reporting database upon request. There may come a time where a more automated approach will be considered, but for now it requires some dialog with the consumer and support.

Please do not hesitate to *contact support* and request access.

Hosted Connections with Power BI

An alternative approach to directly connecting to the database, is to build a report/visual/dashboard using Power BI, and upload it into INFO-Subscription for processing.

However to do that, Power BI needs a model, which is most easily obtained by a direct SQL connection.

For now that means that to enable Power BI building, a regular SQL connection should be established first. Refer to the above section for the details.

2.22 Reporting Datamodel

The following contains a description of the various tables in the reporting database. Each section of the model is organized into seperate areas/silos mostly to indicate their usage area. Though its up to you to be creative.

The datamodel documentation is divided into a section for each of the various areas, as well as sub-sections for each table.

Some of the tables have noteworthy columns that are called out separately, in their respective sections.

2.22.1 General Purpose

The main area is an catch-all where things that are not naturally in one area or another is created. This contains reporting management data, as well as cross cutting data such as Organizations and Subscribers.

Schema Name : dbo

Available Tables

- Organizations
- Products
- Subscribers
- SubscriberContacts
- Tag
- ReportingGroups
- ProductReportingGroups

Organizations

Information about organizations such as organization name, e-mail, phone and address. Organization data is commonly denormalized.

Products

Contains information about all available products.

Noteworthy columns

ProductCategory: This is a grouping field only for reporting and analytics purposes, can be used to split the reports into groups. You create a group, and then link products to the groups via the API.

Subscribers

The main list of all subscribers in the system as well as the details of the primary contact. This information is mainly used for invocing, reporting and deliveries.

Noteworthy columns

ContactId: This is a reference to the subscriber contact from which the contact information is retrieved.

ExternalId: External reference as defined by a third party. Typically CRM or a master data system of sorts.

SubscriberContacts

All contact information for all subscribers, such as the primary contact address, billing addresses and delivery addresses.

Noteworthy columns

IsPrimary: Indicates the contact information is the default that will be used for billing, deliveries etc if not overridden by a specific selection of another contact. **Identification**: May contain information about organization number, customer references and more. Data is stored as a JSON array, and T-SQL JSON Functions can be used to parse the data.

Tag

Tags are text fields that can be associated with different entities, used for grouping and reporting purposes. They carry no value and are only represented in the reporting datamodel.

The idea is to let third parties tag on information as appropriate for their specific use case.

At the current time its possible to provide tags for:

- Subscriptions
- Subscribers
- Subscriber Accounts
- Payments
- Payment Demands

At the current time the API endpoint for creating a new Order contains a tag value that will be assigned to the first subscription, the subscriber and the subscriber account. It is designed as a list of tags where all tags will be added.

The *ReferenceId* will then correspond to *Subscriber*, *SubscriberAccount* and *Subscription*, respectively.

An example is to tag all orders with a “Sales Unit” or “Store” Id, allowing to report and analyze sales by physical department or store.

The vast majority of the entities in the reporting model have a connection to *SubscriberId* and can then be grouped / filtered in relation to the *Tag* given.

ReportingGroups

Reporting groups, as the name suggests, a way to group things for reporting purposes. Specifically Products and permanent discounts. For instance you may want to group all products of a specific type as “Cosmetics” and another as “Foodstuffs”. You can do this in each and every report you generate, or you can setup the group in the reporting model, and just group and filter on the reporting group.

Items can be included in multiple groups.

2.22.2 Economy and Billing

The economy and billing area/silo, contains data related to the recurring billing of the subscriptions. Anything you need to know about your billing is collected here such as Invoices and their state, outstanding transtractions and fees.

Schema Name : economy

Available Tables

- AccountTransaction
- Invoices
- PaymentDemands
- PaymentDemandFees
- PaymentDemandDetails
- PaymentDemandAllowances
- PaymentDemandCharges
- SubscriberLedgers
- SubscriberAmounts

AccountTransaction

Outstanding account transactions. This contains the current list of transaction that has yet to be billed (put on an Invoice). These are generated as payments are processed, subscriptions are cancelled and related.

Invoices

Summary data about the various invoices produced, such as state and issue date. Does not include the entire invoice, just summary data.

PaymentDemands

Invoicing of subscribers produces a "Demand for Payment", and PaymentDemand, along with its related entities, contains details about what is billed such as:

- Reference to the subscription period(s) billed and the amount.
- Any transactions caused by metered billing or manual corrections.
- Time of issuing.
- Reference to the resulting Invoice Document.
- Reference to the Order/first period and the amount.
- State of settlment.

Payment Demand is the basis for payment claims that the system generates, and it is on the basis of these that an invoice is formed.

Example reporting/analysis use cases include:

- Total outstanding dept amount.
- Billed Amount by month or even by product.

- Degree of payment by area, organization or payment method.

PaymentDemandFees

Fees related to payment demands and reminders for the given payment demand.

PaymentDemandDetails

Details refers to the Subscriptions, Orders and external one-time transactions associated with a payment demand.

Keep in mind that Subscriptions are always billed in advance, but the subscription that is referred here is the past one (the one deciding how the next bill is going to be basically).

Noteworthy columns

NextSubscriptionId: This column is populated once the subscription is renewed/extended. Thus at the time when the NextSubscriptionId is populated the demand now covers and existing

PaymentDemandAllowances

Any allowances consumed from the Billing Account (*AccountTransaction*) are presented here. Allowances reduce the total amount to be claimed.

PaymentDemandCharges

Any extra charges/debits from the Billing Account (*AccountTransaction*) are presented here. Charges increase the total amount to be claimed.

Typically these would be metered charges, or left overs from previously settled demands with a missing amount.

SubscriberLedgers

The entity contains all ledger entries/transactions regardless of organization or state.

Represents the current total state of a subscribers accounts.

KPI: SubscriberAmounts

This is the sum of subscriberledgers grouped by subscriberid, organizationid and currency at any given time.

Can be used to give a “snapshot” insight into how much is billed or owed across all organizations. It is designed as a pre-calculated KPI.

2.22.3 Orders

The orders area/silo, contains data related to incoming orders, and basically should be your goto area for sales performance data.

Schema Name : order

Available Tables

- OrderAmounts
- OrderCompletedAmounts
- Orders
- Products

KPI: OrderAmounts and OrderCompletedAmounts

The tables *OrderAmounts* and *OrderCompletedAmounts* are KPI-like tables providing a summary of the order value for a given date and organization.

One table includes all orders that have been started (and later cancelled), while the other only summarises orders that have been completed.

Example use cases involves a running dashboard of the value of your orders over time.

Orders (and Products)

Summary data about all orders started, cancelled and completed.

It includes information such as the price/value, when it was created and completed, as well as when it was abandonned/cancelled if that was the cause.

It refers to a Products table so its possible to group Orders and Products and analyze sales behaviour using this data.

2.22.4 Subscriptions

The subscription area/silo, is similar to the Orders silo, except it related to data about the recurring business, it should be your goto area for reucrring performance data.

Schema Name: subscription

Avtailble Tables

- Subscriptions
- CanceledSubscriptions
- Contracts
- Enterprise Plans
- SubscriptionPackages
- SubscriptionPacakgeProducts

Subscriptions

All subscriptions registered in the system, with comprehensive information about, start and end time, details such as price, tax (VAT), number of units, cancellation status and reason for cancellation as well as renewal status.

CanceledSubscriptions

All canceled subscriptions, including reason for cancellation.

Contracts

Information on subscriptions with fixed contracts, and how long those contracts are defined to last.

EnterprisePlans

General information about agreements for enterprises used for common billing and rules for selling to these enterprises.

SubscriptionPackages and SubscriptionPacakgeProducts

Contains instance level details about a given subscription, detailing the specific for that subscription. More commonly known as a Subscription Plan.

Noteworthy columns

BillingFrequencyId: The frequency at which the Subscription is billed/renewed

- 1001 - Month
- 1003 - Quarter
- 1012 - Full year.

SubscriptionPackageChainId: Defines the package chain for stepping/changing packages during renewal. I.e. first pay 99 kroner the first month, then 149 for the second, 199 for the third before finally changing to the full price of 249.

InitialTermType: This is used if the first period is to have a different length. 10 - «Until date », 20 - «Number of days», 100 - «Out the month», 200 - «Out the year» .

InitialTermValue: This will then have slightly different values depending on the type. For the value «10», then there will be a date. For "20" it will be a number of days ". For "100" and "200", it is not used.

AutomaticStop: This means that the subscription will be automatically stopped after the period.

2.22.5 Payments

Schema Name: payment

Avtailble Tables

- DailyPaidAmounts
- Payments

KPI: DailyPaidAmounts

KPI style table with total amount of what is paid for any given date, for each organization.

This only includes payments that have been approved and complete, payments yet to be identified are not included in the summary.

Payments

All payments for each individual subscriber, regardless of source. The most common source values are OCR, PayEx , Manual, Import and MI (migrated).

2.23 Support and General Inqueries

Priority questions and issues should be directed at the regular INFO-Subscription support channel as described on https://www.infosoft.as/contact-us/. If you are a third party, i.e. not the one with an account/tenant for INFO-Subscription, then you should contact your client and have them facilitate communications.

2.23.1 Reporting Bugs

It should come as no suprise that INFO-Subscription is not without bugs.

This section contains a few pointers on how to submit bug reports and what we would like you to include in said reports.

Bugs can be reported via GitHub if you have a purely technical issue, such as an undocumented response, an invalid response for an action or whatever. We do not guarantee any response time or SLA on the GitHub issues, but we will try and address issues as best we can when we have the time.

For priority bugs please submit them through the regular support channel as well (perhaps just with a reference to the GitHub issue) and the normal SLA applies.

Information in Bug Reports

Please include the following in your bug report

- The Tenant Id/Name
- The Client Id used to generate the request
- A description of the actual behaviour and the expected behaviour
- The raw HTTP request and response that caused the bug

2.23.2 Feedback, Comments and Suggestions

If you want to provide suggestions and general feedback regarding the API, please reach out to us using the GitHub repo.

If you want to request a new feature in the service, the github repo is not really the channel for it.

2.24 Contributing to the Documentation

If for some reason you are compelled to write better documentation we accept your contributions.

The simplest way to contribute is to create a pull request on the documentation source GitHub repo

Or if you discover a documentation error/typo or other iusse, and can't be bothered to fix it, the please open an issue at the same repo so we might get around to fixing it.

2.24.1 Want to contribute but have no idea what to write?

While Infosoft is most likely better equipped to write general documentation on what the systems does and how to use it, we lack the clarity that not knowing a lot of system details gives.

So if you are looking for something to contribute, please consider writing how-to's or turotial style guidances that solves a specific use case.

Examples could be something like:

- How to synchronize Customer data with Microsoft Dynamics
- Getting started with external payments

If you want to know if we are interested in a specific topic or area before you start, then open an issue, and we will reply ASAP.

2.25 Release April 15th 2024

"ABC 123", this release numbers 123 since the inception of INFO-Subscription, as so often before with bug fixes and minor improvements spread throughout the platform.

2.25.1 API and Backend

The following new fixes, features and improvements are now available in the API and the backend.

Added

- Business audit for crediting of payment demands.
- **[Preview]** VippsMobilePay API adjustments for Orders and Agreements with support for MobilePay Denmark and Finland.
- **[Preview]** Proration support for Edition/Issue based frequencies on cancellations.
- Country Code for VippsMobilePay Account Configuration, mostly affects branding (Vipps or MobilePay) today.

Changed

- Deliveries are no longer generated for subscriptions which are entirely in the past (mostly affects migrations).
- Upon cancellation of a Subscription, any scheduled changes are now removed automatically. Restarts will have to reschedule changes. This only affects cancellations not plan changes.
- It is no longer possible to add an Invoice Contact or change Payment Agreement on Subscription belonging to an Enterprise Plan. It had no effect previously, but was the cause of some confusion.
- Up to date product list prices are now calculated when getting Subscription Plans.

Fixed

- An issue where an immediate change of SubscriptionPlan would not supply the correct cancellation type to the subscription.
- An issue where crediting and reissuing of an Enterprise Plan demand would sometimes lead to a missing period for new orders yet to be billed.
- An issue where a reminder on an Enterprise Plan would be reported to belong to the latest issued invoice on the same plan, instead of the one actually being reminded.
- An issue where Enterprise Plan demands would be generated a day off for cases near the DST switch. Now it correctly calculates based on the configured timezone for the issuing organization.
- An issue where deliveries might be split even though no address change occurred. In turn this would lead to surplus notifications to Distribution Innovation or other third parties.
- An issue where it was impossible to delete Denial Orders in case there were multiple registrations on the same subscriber.
- Fixed an edge case where the end calculation for Subscriptions would be invalid if the end time was supposed to be exactly on the switch to DST.
- **[Preview]** An error with processing of BetalingsService mandates and payments during file import.

Deprecated

We call out the following deprecations, the endpoints/field will be removed in a future version, the exact timing depending on existing use and adoption of replacements.

- We are deprecating the endpoints `/paymentdemand/{id}/credit/completely` and `/paymentdemand/{id}/credit/partially` in favor of `/paymentdemand/{id}/credit` endpoint.
- We are deprecating the field `TaxPercentage` on the endpoint for creating Account Payment Demand transaction `POST /paymentdemand/`, in favour of specifying a list of Tax Details.

2.25.2 Merchant Client

The following new features and improvements are now available in the merchant client.

Added

- Option to delete payments that have yet to be completed.
- Option to sort subscriptions on Enterprise Plans based on number or name.
- Filtering for subscriptions on Enterprise Plans.
- Requirement for additional information based on chosen Payment Agreement, specifically for Email and EHF.
- Quick link for unmatched payments from the subscriber payments view.
- **[Preview]** View to configure BetalingsService integration.
- **[Preview]** Subscriber Payment Agreement view for BetalingsService.

Changed

- During Cancellation: Removed checkbox for calculating contract fee if no contract is present.
- Payment Management is redesigned with a new tab order and refreshed grids/list.

Fixed

- An issue where it was not possible to credit a payment demand without reissuing for AvtaleGiro, AutoGiro and BetalingsService.
- An issue where it was was not possible to credit AND reissue Account Payment Demands.
- An issue where listing domain users would fail in case one of the underlying users was removed from the source.

2.25.3 Self-Service and Sales Poster

The following new features and improvements are now available in the self service and sales poster.

Added

- **[Preview]** View for BetalingsService payment agreements.

Fixed

- An issue where existing shared users ordering a new subscription would lead to error messages.
- An issue where adding the last shared users was not allowed (counter error).

2.26 Release March 12th 2024

In addition to general bug fixing and minor improvements we continue to march on with components to do billing based on Editions, the initial feature is mostly ready, with the only notable exception being a lack of automatic proration on cancellations.

This release also extends the preview for BetalingsService payment agreements with additional feature areas and fixes.

2.26.1 API and Backend

The following new fixes, features and improvements are now available in the API and the backend.

Added

- Support for directly utilizing the Infosoft Partner credentials in Vipps communication, added as an option on the Account configuration - resulting in simplified setup/onboarding of new Vipps Sales Units.
- **[Preview]** Support for configuration BetalingsService accounts, using Infosoft as the Data Supplier.
- **[Preview]** Support for importing Payments and Mandates files from BetalingsService.
- **[Preview]** Support for exporting requests for Payments files to BetalingsService.
- **[Preview]** Support for adding subscription plans for products using editions.
- **[Preview]** The order endpoint now understands products and packages configured with Editions, and added support for indicating the number of editions during registrations.

Changed

- **[Preview]** Subscriptions on editions can now be created and renewed like regular time based subscriptions.
- The automated AvtaleGiro mandate lookup is now set up to query multiple in case of intermittent issues with the Mastercard Payment Services API. The change should be mostly transparent as mandates are already imported multiple times each day.
- Calendars now only operate with Dates and not a combination of Date and Time, since the calendar is supposed to present entire days, not time ranges.
- **[Preview]** Allowed switching to BetalingsService Payment Agreements on subscriptions.
- **[Hotfix]** Changed handling of payments such that an exactly matching External Invoice Identifier incoming on a wrong organization is moved to the matching organization invoice. The source of these payments are typically handcrafted files, poorly written integrations, or previous misconfigurations.

Fixed

- Distribution Innovation file export timestamps now include timezone information.
- Missing *UserAuthorizationGranted* and *UserAuthorizationRevoked* events when connecting users to subscribers or subscriber accounts.
- An error where manually defining a new due date during credit with a replacement request would be ignored, leading to an invalid due date for the replacement demand.
- An issue where reminder fee would not be included in the Invoice Reminder for Enterprise Plans.

- An issue where Vipps agreements could be imported without currency.
- An error where rare circumstances could lead to the same demand reminder being generated multiple times.
- A rounding error for invoices leading to invalid EHF files.
- **[Hotfix]** An error with multiple capture attempts on orders with Swedbank Pay in case of intermittent errors from the API.
- **[Hotfix]** A regression for capturing payments for Swedbank Pay where the description extended above the 45 chars limit defined by Swedbank Pay.
- Fixed an issue that would lead to Power Bi datasets (and reports), to not be automatically refreshed, causing wait times for users when opening the reports every day.

Deprecated

We call out the following deprecations, the endpoints/field will be removed in a future version, the exact timing depending on existing use and adoption of replacements.

- We are deprecating the endpoints `/paymentdemand/{id}/credit/completely` and `/paymentdemand/{id}/credit/partially` in favor of `/paymentdemand/{id}/credit` endpoint.
- We are deprecating the field `TaxPercentage` on the endpoint for creating Account Payment Demand transaction `POST /paymentdemand/`, in favour of specifying a list of Tax Details.

2.26.2 Merchant Client

The following new features and improvements are now available in the merchant client.

Added

- Exposed configuration for how to calculate VAT on billing fees.
- Additional validation to allow switching payment agreement to EHF or E-mail.
- Search for ADB2C users based on email address.

Changed

- Vipps setup to support use of Infosoft Partner credentials.
- Switching Subscription Plan now displays the total amount of the new plan.

Fixed

- A performance issue with payments in the overview dashboard to reduce load times.
- An issue where invalid export time was shown for Distribution Innovation (related to missing timezone in the API/Backend).
- An issue where it was not possible to switch to existing AvtaleGiro or eFaktura Payment Agreements.
- An issue where expired products were available for selection during creation of new subscription plans.
- A calculation issue in the dashboard which would include not-completed orders in the totals.

2.26.3 Self-Service and Sales Poster

No subscriber/user facing changes to self-service or the sales poster for this release.

2.27 Release Feburary 6th 2024

We are back with another release.

This time around we have started work on a new setup for billing based on Editions or Issues, or more specifically based on a set of dates in a calendar. Underneath the covers it is still mainly based on time. The feature is not complete, but the changelog will start to reflect that we are introducing parts for this new billing concept.

As usual we have a bunch of minor adjustments and bug fixes to make your recurring billing experience smoother.

2.27.1 API and Backend

The following new fixes, features and improvements are now available in the API and the backend.

Added

- It is now possible to define a Period on Account Demand transactions (lines), these are optional and the Start time will still be derived from the issue time in case they are not given.
- **[Preview]** Support for creating BetalingsService payment agreements and mandates. BetalingsService is the Danish Direct Debit solution similar to the Norwegian AvtaleGiro and Swedish AutoGiro. Just like AvtaleGiro, it is mostly an out-of-band solution that requires little in terms of direct interaction.
- **[Preview]** Support for adding product pricing based on a number of editions.

Changed

- Added new CancellationType for cancellations related to deletions (i.e. subscription that start in the future and are deleted and not just cancelled). This allows clients to do special logic already at the time of cancellation, and most importantly allows them to skip a lookup step which is going to fail since the subscription is now gone (deleted).
- Getting an existing subscription or calculating the next subscription period will now include *Product Name* and *Product Description* for each product in the Subscription Plan.
- Credit Note Id is now forwarded to the Ledger Entries when generating Credit records via Billing, ie. when crediting a payment demand or during automatic crediting for cancelled subscriptions.
- Original AccountTime is now available on Charges and Allowances even when they have been included on a Payment Demand.

Fixed

- **[Hotfix]** An issue where Zip Code would be set with an invalid value during eFaktura agreement scanning.
- **[Hotfix]** An issue where eFaktura agreements would be re-created even though they existed in cases where there was also an AvtaleGiro mandate.
- **[Hotfix]** An issue in the reporting subsystem where replacement Payment Demands would not be generated in the reporting model due to an invalid configured constraint.
- An issue with Account PaymentDemands where the *ProductId* would not be populated all the way through reporting and in the API on the TaxDetails collection.
- An issue where Account PaymentDemands with a Payable Amount of 0 would not be automatically settled.
- An issue where Enterprise Plan Payment Demands would not carry over Product Name and Description for TaxDetails in reporting.
- An issue where adjusting the Billing Plan Minimum Due Days would not affect existing Enterprise Plans. **Note** By design it still does not reschedule anything, only affects new billing cycles and orders.
- An issue where crediting with a replacement payment demand on Enterprise Plans would lead to the subsequent reminder would never be generated.
- A decimal rounding issue on EHF when utilizing multiple non-zero VAT products on a single transaction.
- An issue with failed Template Plan validation where Product, Price and Plan had a start time in the future, causing the plan creation to fail.

Removed

- Invalid Calendar endpoints that were not in use.

Deprecated

We call out the following deprecations, the endpoints/field will be removed in a future version, the exact timing depending on existing use and adoption of replacements.

- We are deprecating the endpoints `/paymentdemand/{id}/credit/completely` and `/paymentdemand/{id}/credit/partially` in favor of `/paymentdemand/{id}/credit` endpoint.
- We are deprecating the field `TaxPercentage` on the endpoint for creating Account Payment Demand transaction `POST /paymentdemand/`, in favour of specifying a list of Tax Details.

2.27.2 Merchant Client

The following new features and improvements are now available in the merchant client.

Added

- **[Preview]** New dashboard/overview visuals to replace the current numbers based overview.
- Organizations can now be configured with Identifications such as Organization Number and other external unique Ids. This was previously removed, but has now been brought back in an updated version.

Changed

- HPR Number can now be configured on a Tenant by Tenant basis, and its toggled off by default.
- Removed Distribution Innovation Endpoint references since they are now determined by the backend.
- Scheduled Plan changes are not shown on cancelled subscriptions, as they are never going to take effect.

Fixed

- An issue where Email Domain users would be listed with the previous period for Subscription Plan access instead of the current period.
- An issue where the Payment Import log would display the Amount for parsed transactions in the processed transactions Amount. Leading to some confusion on OCR files with multiple assignments.
- An issue where displaying an old invoice with a deleted Invoice Contact would lead to an error message.

2.27.3 Self-Service and Sales Poster

The following new features and improvements are now available in the self-service client

Changed

- **[Hotfix]** Salesposter order process URLs are now significantly shorter.

2.28 Release January 4th 2024

Happy new year to everyone.

We start the new year with a new release. This one mostly contains bug fixes and minor adjustments to existing behaviours.

2.28.1 API and Backend

The following new fixes, features and improvements are now available in the API and the backend.

Changed

- The integration setup for Distribution Innovation has been simplified so that URLs are now determine automatically during deployment and configuration of the URL is no longer possible.
- Multiple improvements to the address splitting routine for automatic address lookups using Kartverket. Improves precision for multiple edge cases such as names being partial matches in longer names.
- eFaktura Scanning now skips using known invalid phone and zip codes during agreement lookups.
- If a subscription has an invoice contact, it will no longer be automatically switched to an eFaktura payment agreement.
- Updating out-of-band Payment Agreements (such as eFaktura and AvtaleGiro) now triggers automatic switch of Payment Agreements on running subscriptions.

Fixed

- An issue where order demand schedules would remain even if the resulting subscription was deleted.
- An issue where payment stop would not act on the first subscription period.
- Marking billing account transaction as included in the ledger in all cases.
- An issue where some charges and allowance transactions would not be reverted during automatic crediting (from cancellations).
- An issue where the API did not populate tax details on Enterprise Plan payment demands.
- An issue leading to multiple eFaktura Agreements with the same identifier.
- An issue where invalid EHF files would be produced when a transaction did not have an End Time.
- An issue where the first file to be imported after a release or catastrophic failure would hang until processed manually.
- An issue where scheduled Subscriber contact changes would include a blank newline at the end.

Removed

- It is no longer possible to do file based settlement for Vipps charges. It has not been exposed in the UI for some time, and has been replaced by the automatic API based settlement.

Deprecated

We call out the following deprecations, the endpoints/field will be removed in a future version, the exact timing depending on existing use and adoption of replacements.

- We are deprecating the endpoints `/paymentdemand/{id}/credit/completely` and `/paymentdemand/{id}/credit/partially` in favor of `/paymentdemand/{id}/credit` endpoint.
- We are deprecating the field `TaxPercentage` on the endpoint for creating Account Payment Demand transaction `POST /paymentdemand/`, in favour of specifying a list of Tax Details.

2.28.2 Merchant Client

The following new features and improvements are now available in the merchant client.

Added

- UI Support for additional event types on the trigger setup
  - SubscriptionDeactivated
  - ReminderIssued
  - UserAuthorizationGranted
  - UserAuthorizationRevoked
- The ledger summary total now shows outstanding transactions (yet to be Invoiced), if ledger billing is enabled.
- Added support for showing transfer transactions on the account.
- Support for registering Email and EHF as payment agreements during Orders.

Changed

- AutoGiro crediting now longer displays an option for specifying a custom due time when re-issuing is requested.
- Minor translation and general language improvements.

Fixed

- An issue with missing organization information on the account transactions.
- An issue with the reminders filter where “All states” would end up displaying nothing.
- An issue where filtering grid data would have no effect on the exported file.
- An issue where the default Currency symbol was not applied on invoice details/lines.

2.28.3 Self-Service and Sales Poster

The following new features and improvements are now available in the self-service client

Fixed

- An issue where the default Currency symbol was not applied on invoice details/lines.

2.29 Release November 27th 2023

This release is a bit light on the feature side, however lots of stuff is going on behind the scenes to improve stability, resilience, security and performance.

However a few existing features are now exposed in the Merchant UI.

2.29.1 API and Backend

The following new fixes, features and improvements are now available in the API and the backend.

Added

- Automatic cancellation/deactivation of Vipps Payment Agreements when a cancelled subscription has ended.
- Introduced support for semi-atomic transferring of billing account transactions between billing accounts, and between subscribers. This includes a transfer log for reporting and analytics.

Changed

- Recurring/Agreement payments on Swedbank Pay now supplies the final VAT Amount of the transaction. This is only a reporting value, and only sparingly available in the Swedbank Merchant Portal.

Fixed

- An error where credited and re-issued Payment Demands would not be exported for AutoGiro (Bankgirot Sweden).
- An error which made it impossible to cancel/remote AutoGiro mandates

Deprecated

We call out the following deprecations, the endpoints/field will be removed in a future version, the exact timing depending on existing use and adoption of replacements.

- We are deprecating the endpoints `/paymentdemand/{id}/credit/completely` and `/paymentdemand/{id}/credit/partially` in favor of `/paymentdemand/{id}/credit` endpoint.
- We are deprecating the field `TaxPercentage` on the endpoint for creating Account Payment Demand transaction `POST /paymentdemand/`, in favour of specifying a list of Tax Details.

2.29.2 Merchant Client

The following new features and improvements are now available in the merchant client.

Added

- Option to toggle whether to reduce reminder amounts with surplus accounted payments (allowances).
- It is now possible to cancel Vipps Payment Agreements.
- When ordering from merchant it is possible to choose payment types besides Invoice.
- Adding an Organization Number validation for tenants with Norwegian organizations when attempting to switch/use an EHF Payment Agreement.
- Introduced option for quick navigation to Enterprise Plan owners from the Enterprise Plan configuration.

Changed

- Exposed existing Payment Provider Types for EHF and Email in various workflows.
- All credit operations now go through the new endpoint and credit flows will transfer back account transactions as mentioned in the previous release.
- Removed/Sanitized the Invoices and Payments grid in the subscriber overview.
- Refunds are now highlighted with a separate colour.

Fixed

- Removed possibility to delete Enterprise Plan owners with active enterprise plans.
- An issue where Cancellation Cause would disappear when changing a previously registered cancellation.
- An issue where Invoice/CreditNote copy on Enterprise Plan would not redirect back to the details.

2.30 Release October 31st 2023

Only a 3 week release cycle this time around, mostly because the previous release was a bit delayed. Just a few items in this release, mainly regular maintenance and small improvements, but a few quality of life improvements for Merchant and Billing has been included as well.

2.30.1 API and Backend

The following new fixes, features and improvements are now available in the API and the backend.

Added

- OpenAPI/Swagger specification now include security information on all endpoints requiring it. Previously only a global declaration and a few exemptions were present.
- OpenAPI/Swagger specification now includes default responses for most endpoints. This to ensure better compatibility with code generation tools.
- A new external event type *SubscriptionDeactivated* is now available, it triggers when the subscription actually expires. Refer to the event documentation for more details.
- Support for manually deactivating a Vipps recurring agreement. Specifically removes the subscription in the Vipps app.

- Automatic de-activation of Vipps recurring payment agreements when the abstract Payment Agreement is deactivated or deleted. See previous item.
- Billing Account Transactions now contains a property *IncludedInLedger* defining if it already mirrored in the Subscriber Ledger or not. This property is also present when creating/generating account transactions.

Changed

- EHF serialization process re-implemented. Will be rolled out on tenants progressively.
- Invoice Documents now group tax totals. Simplifies generation of third party invoices such as PDFs, Emails, eFaktura and EHF.
- Manually crediting Payment Demands will now revert account transactions to the billing account where they originated. Mainly to ensure allowances/additional payments are not written off, without explicit declaration.
- Generation of reminders will now only consume allowances generated by payments and not arbitrary allowances.

Deprecated

We call out the following deprecations, the endpoints/field will be removed in a future version, the exact timing depending on existing use and adoption of replacements.

- We are deprecating the endpoints `/paymentdemand/{id}/credit/completely` and `/paymentdemand/{id}/credit/partially` in favor of `/paymentdemand/{id}/credit` endpoint.
- We are deprecating the field `TaxPercentage` on the endpoint for creating Account Payment Demand transaction `POST /paymentdemand/`, in favour of specifying a list of Tax Details.

2.30.2 Merchant Client

The following new features and improvements are now available in the merchant client.

Added

- New view for bulk registration of Subscribers and Subscriptions for Enterprise Plans.

Fixed

- Registration of subscriptions on Enterprise Plans, now only show subscription plans that matches the billing alignment of the selected Enterprise Plan.
- Translations on Payment Agreement details in Norwegian.
- An issue where it was possible to add prices with an invalid start time.

Changed

- Removed option for adding an Invoice Contact when registering subscriptions on an Enterprise Plan.
- Reworked the Enterprise Plan details view with new grid components, and adjusted filtering options and placement order.

2.30.3 Self-Service and Sales Poster

The following new features and improvements are now available in the self-service client

Fixed

- An issue where it was impossible to remove Organization Number and Invoice Reference on a subscriber contact.

2.31 Release October 10th 2023

This time around we have been hard at work to iron out issues with deliveries, as well as some other features and bugs here and there. Due to autumn vacations the release has been a bit delayed.

2.31.1 API and Backend

The following new fixes, features and improvements are now available in the API and the backend.

Added

- New SwedbankPay API endpoint. This is just a re-branding of the old PayEx and PayEx eCommerce endpoints.
- Added option for deleting Organizations from the API.
- An optional failure state and reason on processed payment requests handled by the Swedbank Pay integration (Card payments).
- Added an option to update a few details on a SwedbankPay Account configuration, such as the encryption key.

Fixed

- An issue where an unknown organization id would cause an internal server error, it now correctly produces a client error (400 Bad Request).
- An issue where Enterprise Plans with many subscriptions (150+) would end up not being settled properly when processing payments.
- An issue where future orders on eFaktura and AvtaleGiro, would have a missing date part in the statement text for the period.
- An issue where an unknown organization id would cause an order processing failure instead of a client error.
- An issue where product authorization would not be properly deleted on deleted users. No authorization was granted, but the record of previous authorization was kept indefinitely.
- Multiple issues related to Deliveries and the associated Distribution Innovation integration, including:

  - Mixing delivery stop and temporary address changes would lead to invalid or missing restarts.
  - Manual use of the "Source" field would cause processing failures if no DI integration was configured.
  - Delivery stops with a delivery restarts overlapping a cancelled subscription could lead to deliveries being started on cancelled subscriptions.
  - Updating an address on subscription end would register an invalid delivery split.

Changed

- Payment Demand Charges and Payment Demand Allowances now include the original transaction type as it was on the account.
- Empty address lines on Subscriber Contacts are now stripped away during updates and creation.
- Payments that are processed on a subscriber with only one Subscriber Account will now be automatically associated with that account.
- Improved the mapping of Vipps recurring agreements when querying for the remote status of a charge.

Deprecated

We call out the following deprecations, the endpoints/field will be removed in a future version, the exact timing depending on existing use and adoption of replacements.

- We are deprecating the endpoints `/paymentdemand/{id}/credit/completely` and `/paymentdemand/{id}/credit/partially` in favor of `/paymentdemand/{id}/credit` endpoint.
- We are deprecating the field `TaxPercentage` on the endpoint for creating Account Payment Demand transaction `POST /paymentdemand/`, in favour of specifying a list of Tax Details.

2.31.2 Merchant Client

The following new features and improvements are now available in the merchant client.

Added

- Support for adding scheduled contact changes, such as a future address change.
- Adding an Subscriber Organization Number will now verify it against the PEPPOL database for validity in terms of receiving EHF invoices. A visual clue will be displayed if it does not exist.
- Added validation for delivery changes to allow processing on the next day at the earliest. Changes further into the future are still accepted.

Fixed

- The Distribution Innovation overview now allows omitting integration values that are not available.
- Delivery changes are now stored with midnight as the time of change instead of the current time part.
- An issue where an invalid amount was shown for reminders.
- An issue where the payment registration view would not always reset the input fields as required.
- An issue where reminder settings order would not be persisted.

Changed

- Replaced the grids for Invoices, Payments and the combined view so it now supports paging the results. Reducing the consumption of screen real estate.
- Multiple text adjustments and additions for Payment Agreements.

2.31.3 Self-Service and Sales Poster

The following new features and improvements are now available in the self-service client

Added

- Added validation for delivery changes to only allow processing on the next day.

Fixed

- Support for adding scheduled contact changes, such as a future address change.
- Delivery changes are now stored with midnight as the time of change instead of the current time part.

2.32 Release September 5th 2023

A short sprint this time around, and thus also a minor release. Mostly this release contains bug fixes, but a few new features have been shoehorned in!

2.32.1 API and Backend

The following new fixes, features and improvements are now available in the API and the backend.

Added

- **[Preview]** Added support for automatically settling payment demands and crediting reminder fees when receiving payments on the original demand amount.
- Added new endpoint to calculate the next subscription period for any given subscription, based on the current registered information.
- The eFaktura integration now supplies the issuing organization number during agreement scanning, effectively ensuring that subscribers who opt-out of eFaktura for a given issuer will not be automatically added.

Fixed

- A validation issue where changing the plan of an existing subscription required more options to be set than specified by the documentation and internal requirements.
- The API reference documentation with the correct status code, `HTTP 202`, for a successful change of payment agreement.
- An issue where it was possible to provide an empty/default `UUID` for Invoice Identifier generation configuration.
- An issue where the recently introduced event types for domain authorization would not be published correctly.
- An issue where tenants with a special character in the name could not enable the event notification infrastructure.
- An issue with billing where the payment demand schedule for an order would be deleted in case the order was in the future, and the subscription was configured with automatic cancellation at the end of the first period.

Changed

- Vipps based orders should now trim away white spaces in the phone number, reducing the number of invalid requests sent to Vipps. Vipps have also deployed a change in their validation routine to allow different formatted phone numbers.
- Details on Enterprise Plan payment demands would not be generated with a new Id after partial credits. This has now been changed so that the details are regenerated with a new Id.

Deprecated

We call out the following deprecations, the endpoints/field will be removed in a future version, the exact timing depending on existing use and adoption of replacements.

- We are deprecating the endpoints `/paymentdemand/{id}/credit/completely` and `/paymentdemand/{id}/credit/partially` in favor of `/paymentdemand/{id}/credit` endpoint.
- We are deprecating the field `TaxPercentage` on the endpoint for creating Account Payment Demand transaction `POST /paymentdemand/`, in favour of specifying a list of Tax Details.

2.32.2 Merchant Client

The following new features and improvements are now available in the merchant client.

Added

- Support for restarting a cancelled subscription with the exact same plan as it was on previously. Useful for handling cases with late payments and similar.
- Support for removing invoice contacts as a scheduled change.
- **[Preview]** Connected domain users are now listed on the users list.

Fixed

- An issue where temporary delivery changes would cause a validation error, but the changes would still be partially applied.
- An issue where setting/changing the Kilkaya installation URL would not have any effect.
- Visuals for frequency during order creation in chrome.
- **[Preview]** A validation issue that prevented registration of domain names with a dash (-) in the domain authorization.

Changed

- Replaced the subscriptions grid view on the overview and on the subscriptions tab.
- Multiple text adjustments and additions for Payment Agreements.

2.32.3 Self-Service and Sales Poster

The following new features and improvements are now available in the self-service client

Fixed

- An issue where temporary delivery changes would cause a validation error, but the changes would still be partially applied.
- **[Preview]** An issue where connecting a new user with a domain authorization would occasionally fail to register.

2.33 Release August 10th 2023

Summer vacations are almost over for everyone in the dev team, and a new release is ready, roughly 2 months since the previous release. This release is a mixed bag of fixes, small incremental improvements and a single new feature in preview.

We are still trying out the new format for the changelog, following a type of change convention outlined on https://keepachangelog.com/

2.33.1 API and Backend

The following new fixes, features and improvements are now available in the API and the backend.

Added

- Added new public event types for `UserAuthorizationGranted`, `UserAuthorizationRevoked`, `EmailDomainAuthorizationGranted` and `EmailDomainAuthorizationRevoked`. Enables additional external reactions based on user authorization changes.
- Added support for Tax Details on Charges and Allowances. This means that Charges on Payment Demands will now include a Tax Detail breakdown instead of a single Tax Percent. The net effect being that multiple charges will be merged into a single Charge with multiple taxes in several scenarios.
- Support for removing InvoiceContact via a scheduled subscription plan change.
- Added support for switching Payment Agreement during Subscription restarts.
- **[Preview]** Support for User Email Domain and Authorization mapping.
- **[Preview]** eFaktura Mandate/Agreement Scanning will now be executed on a recurring schedule, with partial scans every day and full scans every week.

Fixed

- Updating an existing BillingPlan without specifying a frequency will now produce a validation error.
- Fixed an issue where Account Payment Demands would issue a reminder with a Payable Amount of 0.00 in cases where an outstanding Allowance should have settled the demand.
- Vipps charges now have a due date the day after the Payment Demand to avoid issues with early cancellations of time based subscriptions.
- Added missing validation for creating multiple AvtaleGiro account setups with the same Bank Account Number.
- Fixed an issue where an additional empty Payment Demand would be generated for Enterprise Plans upon a partial credit.
- **[Preview]** eFaktura transactions with multiple identical Tax Groups will now be bundled together to produce a better visual experience.
- **[Preview]** Fixed an issue where eFaktura transactions would not be generated in case the issuing Organization was missing phone number or contact email.

Changed

- Payment Import API has been slightly modified in terms of the output. The input is unchanged.
- Simplified account configuration for AvtaleGiro and eFaktura. Multiple input fields has been removed (defining them will not break things, they will just be ignored).
- Subscription Restarts will now be routed through the Order processing to facilitate handling of multiple Payment Agreement scenarios.
- Multiple adjustments to the underlying Payment Import file processing, in order to improve feedback and resilience.
- Scheduled Subscription Plan changes will no longer split the existing Subscription in cases where there is no need. For instance when changes are set to affect a future Subscription period.

Deprecated

We call out the following deprecations, the endpoints/field will be removed in a future version, the exact timing depending on existing use and adoption of replacements.

- We are deprecating the endpoints `/paymentdemand/{id}/credit/completely` and `/paymentdemand/{id}/credit/partially` in favor of the afore mentioned `/paymentdemand/{id}/credit` endpoint.
- We are deprecating the field `TaxPercentage` on the endpoint for creating Account Payment Demand transaction `POST /paymentdemand/`, in favour of specifying a list of Tax Details.

[Preview] New Feature : Email Domain based User Authorization

This release introduces building blocks for automating User Authorization based on the domain name of their email address. It is an alternative to *Site Access* where all users created with a specific domain name, will be authorized to a given domain.

It will work side-by-side with other user Authorization mapping concepts, so you can mix and match as you see fit.

This is still in preview, and some of the UI elements are not yet in place. Additionally some external modifications are needed in the IdP and a custom sign-up experience responsible for mapping the user to the domain authorization.

Interested in participating in the preview? Contact support, find the details on {SUPPORTPAGE}.

2.33.2 Merchant Client

The following new features and improvements are now available in the merchant client.

Added

- Merchants can now configure Self-Service behaviour for Subscription Cancellations.
- Added an indicator showing if there are existing notes on a given subscriber.
- **[Preview]** Support for listing the sum of not-invoice Billing Account transactions on the ledger summary.
- **[Preview]** Support for configuring Domain Authorization on Subscriptions.

Fixed

- An issue with the wrong Tax Amount being shown for Reminder and Invoice Fees.
- An issue where product with an adjusted price would be displayed as expired.
- Tenant selection should again be persisted between session on the same browser.
- An invalid Subscription Plan would sometimes be displayed when doing refunds.
- An issue where it was not possible to remove a planned cancellation of an already renewed Subscription.
- Removed an invalid button from eFaktura payment agreement view.
- An issue with custom grid sorting for eFaktura and AvtaleGiro transactions being ignored.
- An issue where the wrong API endpoint was used for Enterprise Plan Payment Demand crediting.

Changed

- Minor Visual adjustments to the following views:
  - Enterprise Plan Details
  - Invoices
  - Logout confirmation
  - AvtaleGiro and eFaktura account setup
- Payment Import has been adjusted to reflect the API changes, and provide better user feedback.
- Adjusted translations for eFaktura and AvtaleGiro agreements
- Exporting Refunds now include the underlying External Reference field.

2.33.3 Self-Service and Sales Poster

The following new features and improvements are now available in the self-service client

Changed

- Translations in relation to Subscription Cancellations.

Fixed

- An issue where temporary delivery changes would not include the existing Subscriber Contact Name in the change.
- An issue where the Product id would be shown instead of the product name on cancelled subscriptions.
- Default Country would not be set when doing temporary delivery changes.

2.34 Release June 13th 2023

Time flies when you are having fun developing subscription software. It's time for another release, the last one before the summer holidays kicks in. That means the next release will not follow our regular monthly cadence - we will be back with another release some time in August most likely.

We will try with a new format for the changelog, it will still be divided into three main parts, but we will also attempt to follow a type of change convention outlined on https://keepachangelog.com/

This release is a mix of bug fixes and some minor new features, the most interesting being a new behaviour for cancelling subscription based on their lack of payments.

2.34.1 API and Backend

The following new fixes, features and improvements are now available in the API and the backend.

Added

- Subscription Payment Demands and Order Payment Demands can now be credited via the POST `/paymentdemand/{id}/credit` endpoint.
- Support for bulk replacing all Reminder Settings on a Dunning Process via the Dunning Process update endpoint.
- Support for adding/replacing Payment Request Settings on a Dunning Process via the Dunning Process update endpoint.
- **[Preview]** New logic for evaluating automatic cancellation due to missing payments (Payment Stop).

Fixed

- Billing Account transactions would not always apply currency to the data model, causing it to be unavailable in the API and reporting/analytics solution.
- Fixed an issue where removing a future cancellation would lead to invalid accounting and ledger information because any prorated charges would be invoiced again and any allowances would be written off/credited.
- Fixed an issue where it was impossible to credit an Order Payment Demand for a future subscription that had been deleted.
- If a billing account had an outstanding Payment Demand with an allowance in excess of the total payment demand amount, and a new payment was registered, the excess allowance would disappear from the account. This issue has now been fixed.
- All Payment Demands are now correctly assigned a time component of the payment due, this fixes an issue where Payment Requests would be captured slightly before the actual due time (Mostly relevant for Swedbank Pay).
- Swedbank Pay agreements are now stored without time component for the expiration to prevent undefined adjustments of the timezone (the timezone is unknown and depends on the card issuer).

Changed

- Payment Requests for Vipps Recurring are now offset by 1 entire day, to alleviate a problem where Vipps captures money earlier than the actual due time (Vipps operates in entire days, while INFO-Subscription operates with a time component).
- Adjusted/Improved the Billing Account selection process When generating new orders, specifically start of a new subscription after a cancellation time has been surpassed but before the original period expired will now reuse the account in case its the only one available. Previously a new one would be generated.
- **[Preview]** eFaktura agreements will no longer be automatically applied to subscriptions with a length of less than 5 days. To avoid them always being paid late.

Deprecated

We call out the following deprecations, the endpoints will be removed in a future version, the exact timing depending on existing use and adoption of replacements.

- We are deprecating the endpoints `/paymentdemand/{id}/credit/completely` and `/paymentdemand/{id}/credit/partially` in favor of the afore mentioned `/paymentdemand/{id}/credit` endpoint.

Security

- Fixed an issue where a compromised upstream service for EHF receiver validation or Azure Event Grid could lead to our services being used for a DoS attack.

[Preview] New behavior: Automatic cancellation for missing payments (Payment Stop)

We have introduced an alternative behaviour to automatically cancel subscription with a lack of payments. With the modified behaviour all payment demands that are due before the threshold will be evaluated and the amount evaluated is the sum of these demands. In addition any outstanding billing account Allowances will be subtracted from the amount.

Only if the final amount is above the balance threshold will the subscription be cancelled.

Interested in participating in the preview? Contact support, find the details on {SUPPORTPAGE}.

2.34.2 Merchant Client

The following new features and improvements are now available in the merchant client.

Added

- **[Preview]** Support for Interactive Distribution Innovation Address Lookup (Address Helper) during address registration.
- Aggregated total of outstanding account transactions from the billing account are now shown in the Subscriber Ledger.

Fixed

- An issue with the total amount on Subscriber Ledger for Enterprise Plans.
- An issue where adding a Vipps configuration would present an error message even though the configuration was created.
- Updating an Organizations main identifier (Organization Number) would reset other identifications configured, this has now been corrected.
- An issue where crediting an Invoice for an Enterprise Plan, with a replacement, would lead to an error when generating the replacement Invoice.
- Contract time was shown incorrectly when changing Subscription Plan.
- An issue where it was not possible to remove a future cancellation.

Changed

- Minor visual improvements to outstanding billing account transactions.
- Multiple improvements and adjustments to the deliveries view.

2.34.3 Self-Service and Sales Poster

The following new features and improvements are now available in the self-service client

Added

- **[Preview]** Support for Interactive Distribution Innovation Address Lookup (Address Helper) during address registration.

Fixed

- When no sales terms are configured, an error message will be displayed instead of causing an internal server error.
- An issue with Orders using Vipps would keep on displaying the message to not close the window after completing the Vipps interaction.

[Preview] Interactive Distribution Innovation Address Lookup (Address Helper)

With the Interactive DI Address Lookup Merchants and Subscribers will be presented with address suggestions based on information from the DI Address Helper API. As the user enters street information, and selects a street additional suggestions will be provided for house, apartments etc.

The end result is a high quality address that is directly transferred to DI when generating Distribution and Delivery information.

Interested in participating in the preview? Contact support, find the details on {SUPPORTPAGE}.

INDEX