INFORMATIONSARK & UDVIKLERDOKUMENTATION

Theta Interoperabilitet

Apps i Theta-grupper og "Log ind med Theta": sådan forbinder foreninger, klubber og fællesskaber deres egne værktøjer med Theta, og medlemmerne bevarer kontrollen over deres data.

Del 1 · Overblik

Hvad interop er, hvad apps kan, hvordan de installeres, og hvordan det ser ud for medlemmerne. Med fiktive eksempler. (Dansk)

Part 2 · Developer docs

OpenID Connect, the App API, cards, webhooks, notifications and membership verification. Reference for external developers. (English)

DEL 1 · OVERBLIK

Theta, forbundet med resten af foreningens værktøjer

Fællesskaber på Theta bruger ofte andre systemer ved siden af: en bookingkalender, en medlemsdatabase, en turlog eller en hjemmeside med et klubhus-opslag. Theta Interoperabilitet lader de systemer arbejde sammen med Theta, så medlemmerne ikke skal have en ny konto, og så det, der sker i appen, kan ses der, hvor fællesskabet allerede er: i gruppen.

Log ind med Theta

En ekstern app eller hjemmeside lader folk logge ind med deres Theta-konto. Medlemmet ser præcis, hvad appen beder om, og siger ja eller nej. Bygget på den åbne standard OpenID Connect.

Forbundne apps

En app installeres i en eller flere grupper. Den kan skrive opslag og kommentarer, vise interaktive handlingsbokse med knapper, sende notifikationer og få besked, når der sker noget i gruppen.

Medlemsverificering

En forening med sit eget medlemssystem kan bekræfte, at en Theta-bruger er medlem. Grupper kan kræve det for at blive meldt ind, eller melde verificerede medlemmer ind automatisk.

Hvad mener vi med ”app”?

"App" kan betyde mange ting. I dette dokument er en app **et andet system, som en forening eller et fællesskab bruger ved siden af Theta**, og som er koblet til Theta. Det kan være en hjemmeside, et webværktøj, en medlemsdatabase eller en telefonapp. Det afgørende er, at den har sin egen adresse og drives af foreningen eller dens leverandør, ikke af Theta.

- **Ikke Theta-appen.** Theta selv findes også som app til telefonen. Den er ikke en "app" i denne betydning.
- **Ikke noget, man installerer på telefonen.** Når vi skriver, at en app "installeres i en gruppe", betyder det, at Theta-teamet giver den adgang til gruppen. Medlemmerne skal ikke hente noget.

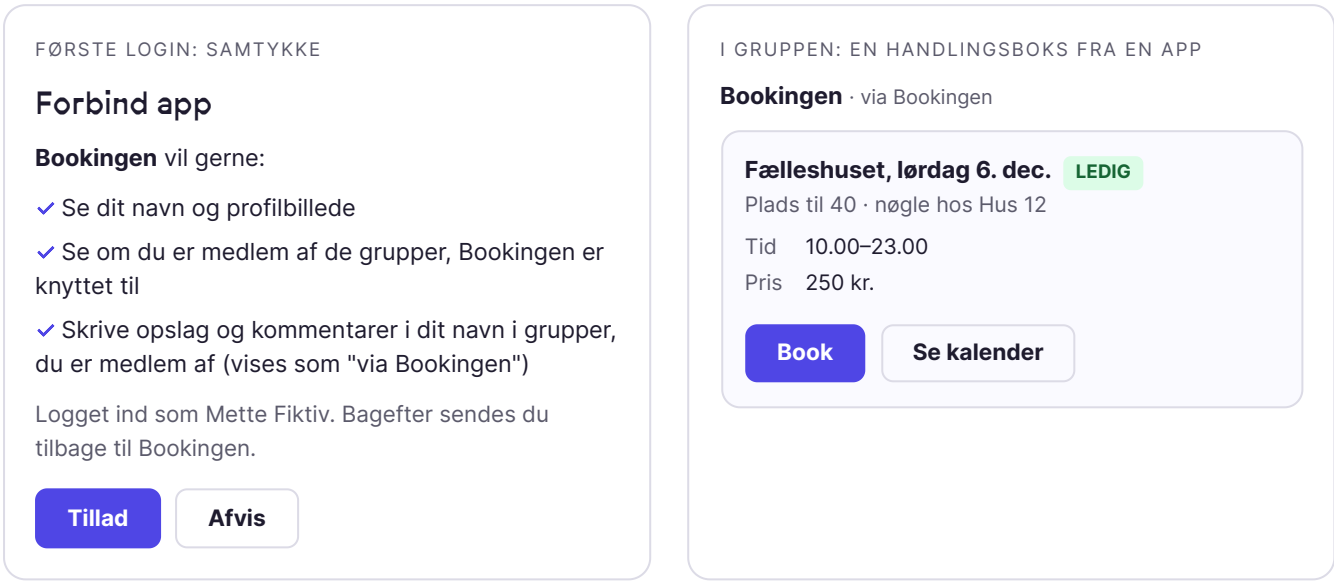
Medlemmerne ser apps ét sted: under *Indstillinger → Forbundne apps*, og som "via <app>" på det, appen har skrevet. Teknisk er en app en registreret OpenID Connect-klient med sin egen bot-konto (se del 2).

Hvad kan en app?

- **Logge medlemmer ind** med Theta og få navn, profilbillede og, hvis medlemmet siger ja, e-mailadresse, verificeringer og medlemskab af de grupper, appen er knyttet til.
- **Skrive opslag og kommentarer** i de grupper, den er installeret i. Enten som appen selv, eller i et medlems navn, hvis medlemmet har givet lov. Alt, appen skriver, er mærket *”via <app>”*.
- **Vise interaktive handlingsbokse** på et opslag: en titel, et mærke (fx "Ledig"), op til seks felter og op til tre knapper. Når et medlem trykker, spørger Theta appen, og boksen opdateres med det samme.
- **Læse gruppens opslag og kommentarer** på et medlems vegne, hvis medlemmet har givet lov, fx for at vise gruppens seneste nyt inde i appen.
- **Sende notifikationer** til gruppens medlemmer eller til folk, der har forbundet appen. De vises i klokken og som push-besked på telefonen.
- **Få besked med det samme** (webhooks), når nogen kommenterer appens opslag, skriver i gruppen, eller melder sig ind og ud.

En app kan **ikke** se grupper, den ikke er installeret i, læse private beskeder, eller handle i et medlems navn uden medlemmets samtykke. Medlemmets skrivninger går gennem gruppens almindelige moderation, og opslag, der venter på godkendelse, forlader aldrig Theta.

Sådan ser det ud for medlemmet



- **Log ind:** Medlemmet trykker "Log ind med Theta" i appen og lander på Theta. Er man allerede logget ind, skal man kun tage stilling til, hvad appen beder om.
- **Samtykke:** Det huskes per app. Beder appen senere om mere, bliver medlemmet spurgt igen.
- **Overblik og fortrydelse:** Under *Indstillinger → Forbundne apps* ser medlemmet alle apps, det har forbundet, hvad de har adgang til, og kan trykke *"Fjern adgang"*. Det, man allerede har skrevet, bliver stående.

Sådan kommer en app i gang

Apps installeres i dag af Theta-teamet. Der er endnu ingen selvbetjening, ingen app-butik og ingen knap for gruppeadministratorer. Det er et bevidst valg, mens funktionen er ny: hver app bliver set igennem, før den får adgang til et fællesskab.

1. **Aftal det med Theta.** Fortæl, hvad appen skal kunne, og hvilke grupper den skal knyttes til. Gruppens administratorer skal være med på det.
2. **Send appens oplysninger:** navn, logo, kort beskrivelse, adresserne, Theta må sende folk tilbage til efter login og logout, og eventuelt en webhook-adresse.
3. **Theta registrerer appen** og installerer den i de aftalte grupper. Appen får en egen bot-konto, som er medlem af grupperne, så den kan skrive som sig selv. I får et client-id; client secret og webhook-hemmelighed udveksles sikkert.
4. **Byg og test** mod Thetas testmiljø efter aftale, og gå derefter i drift på `app.theta.dk`.
5. **Afinstallation** sker også gennem Theta. En app kan midlertidigt slås fra med det samme. Så kan den hverken logge folk ind eller kalde API'et.

Privatliv og sikkerhed

- **Medlemmet bestemmer.** Hver type adgang er en separat linje i samtykket, og samtykket kan trækkes tilbage når som helst.
- **Kun de grupper, appen er installeret i.** Appen kan ikke se andre grupper, og den kan kun handle i et medlems navn i grupper, hvor personen er aktivt medlem.
- **Åbne standarder:** OpenID Connect med PKCE, signerede tokens (RS256) og signerede webhooks (HMAC-SHA256, CloudEvents). Hemmeligheder gemmes krypteret eller kun som en kontrolværdi.
- **Moderation gælder også apps.** Et medlems opslag via en app behandles som alle andre opslag i gruppen. Indhold, der venter på godkendelse eller er skjult, sendes aldrig ud til apps.

Eksempler

De tre eksempler nedenfor er fiktive og viser, hvad man kan bygge. Navnene på foreninger og personer er opdigtede.

FIKTIVT EKSEMPEL

Haveforeningen Solsikken: booking af fælleshuset

Foreningen har en lille bookingside. Med **Log ind med Theta** kan kun medlemmer af gruppen "Solsikken – beboere" logge ind; appen ser det via `groups`. Hver mandag lægger appen et opslag i gruppen med en **handlingsboks** for hver ledig weekend: mærket "Ledig" og knappen "Book". Trykker Mette på "Book", får appen besked, reserverer dagen, og boksen skifter med det samme til "Booket af Mette". Appen sender en **notifikation** til Mette, når nøglen er klar.

FIKTIVT EKSEMPEL

Roklubben Bølgen: turlog i gruppen

Klubbens turlog skriver hver søndag ugens kilometer-top-3 som et opslag *"via Bølgeloggen"*. Når en roer logger en tur, kan appen, hvis roeren har givet lov, skrive en kort kommentar i roerens navn: "Har roet 18 km til Svanemøllen". Gennem **webhooks** får appen besked, når nogen melder sig ind i gruppen "Bølgen – medlemmer", og opretter selv en roerprofil til dem.

FIKTIVT EKSEMPEL

Foreningen Grønne Stier: medlemskab bekræftet automatisk

Foreningen har sit eget medlemssystem. Det kobles til Theta som **verificeringsudbyder**: Et medlem trykker "Bekræft medlemskab" på Theta, logger ind hos foreningen, og får et mærke ved sit navn. Gruppen "Grønne Stier – medlemsforum" **kræver** mærket for at blive meldt ind, og lokalgrupperne melder folk ind **automatisk** ud fra deres lokalafdeling. Melder man sig ud af foreningen, sender medlemssystemet en hændelse, og Theta melder personen ud af de grupper, man kom ind i via verificeringen.

PART 2 · DEVELOPER DOCUMENTATION

Building on Theta

This part is the technical reference for developers of connected apps and of membership verification providers. Everything uses open standards (OpenID Connect, OAuth 2.0, CloudEvents) and plain JSON over HTTPS.

1. Environments and registration
2. Login with Theta (OpenID Connect)
3. Scopes and claims
4. The App API: authentication and conventions
5. App API endpoints
6. Cards (`theta.card/v1`)
7. Webhooks: events from Theta
8. Notifications
9. Membership verification providers
10. Limits and checklist

1. Environments and registration

Environment	Base URL	OIDC issuer
Production	`https://app.theta.dk`	`https://app.theta.dk/oidc`
Test	by arrangement with the Theta team	`<base>/oidc`

There is no self-service sign-up or dynamic client registration. The Theta team registers your app from the information below, installs it in the agreed groups, and creates its **bot account**, the Theta user your app posts as.

Field	Notes
`slug`	Lowercase `[a-z0-9-]` , e.g. `bookingen` . Shown in the member's settings URL.
`name` , `description` , `logoUrl`	Shown on the consent screen, on "via <name>" bylines, cards and notifications. The bot account carries the name.
`clientId`	Your OIDC `client_id` .
client secret	Exchanged out of band. Theta stores only a verifier, so keep your copy.
`redirectUris`	Exact-match list of callback URLs. Use https.
`postLogoutRedirectUris`	Optional; where Theta may send people after logout.
`webhookUrl` + webhook secret	Optional; receives signed events (section 7) and card button presses (section 6).
groups	The groups to install the app in, agreed with the groups' admins.

2. Login with Theta (OpenID Connect)

Theta is a standard OpenID Connect provider. Use any certified OIDC client library and point it at the discovery document; read endpoint URLs from discovery rather than hard-coding them.

Endpoint	Path (under the issuer)
Discovery	`/oidc/.well-known/openid-configuration`
Authorization	`/oidc/auth`
Token	`/oidc/token`
Userinfo	`/oidc/me`
JWKS	`/oidc/jwks`
End session (logout)	`/oidc/session/end`
Token revocation	advertised in discovery as `revocation_endpoint`

Rules of the flow

- **Authorization code flow only** (`response_type=code`), with **PKCE (S256) required** on every request.
- Client authentication is **`client_secret_basic` only**: send the secret in the `Authorization: Basic` header. A secret in the form body is rejected with `invalid_client` .
- The token endpoint has no CORS: exchange codes on your server, never in the browser.
- ID tokens are signed with **RS256**; verify them against the JWKS. Keys can rotate: select the key by `kid` .
- **No refresh tokens.** When the access token expires, send the person through `/oidc/auth` again. With an active Theta session and existing consent, that is a silent redirect.
- Theta pages cannot be framed. Use a full-page redirect or a popup, not an iframe.

Artifact	Lifetime
Authorization code	120 s
Access token / ID token	1 hour
App (client-credentials) token	1 hour
Provider session, remembered grant	14 days

Example

```
# 1. Send the person to Theta (PKCE S256, state and nonce from your session)
https://app.theta.dk/oidc/auth?response_type=code
  &client_id=bookingen
  &redirect_uri=https://booking.solsikken.example/auth/callback
  &scope=openid%20profile%20groups%20posts
  &state=af0ifjsldkj&nonce=n-0S6_WzA2Mj
  &code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM&code_challenge_method=S256

# 2. Exchange the code on your server
curl -u 'bookingen:$CLIENT_SECRET' https://app.theta.dk/oidc/token \
  -d grant_type=authorization_code -d code=$CODE \
  -d redirect_uri=https://booking.solsikken.example/auth/callback \
  -d code_verifier=$VERIFIER

# 3. Read the claims
curl -H "Authorization: Bearer $ACCESS_TOKEN" https://app.theta.dk/oidc/me
```

```
{
  "sub": "48213",
  "name": "Mette Fiktiv",
  "picture": "https://app.theta.dk/…/avatar.png",
  "groups": [
    { "id": "1029", "slug": "solsikken-beboere", "name": "Solsikken – beboere", "role": "member" }
  ]
}
```

If the person presses *Afvis* (deny), you receive `error=access_denied` on your redirect URI. Other errors follow OAuth 2.0 (`invalid_client`, `invalid_grant`, `invalid_scope`, `invalid_token`).

Logout

Send the person to the end-session endpoint with `id_token_hint` and a registered `post_logout_redirect_uri`. This ends their session with your app only. They stay logged in on Theta.

3. Scopes and claims

Scope claims are returned by userinfo only. The ID token carries the standard claims (`sub`, `iss`, `aud`, `nonce`, `auth_time`, …). Each scope is a separate line on the consent screen, in Danish.

Scope	What you get	Consent line shown
`openid`	`sub`: the Theta user id as a string. The same for every app.	(the login itself)
`profile`	`name`, `picture` (absolute URL; a letter avatar when the person has no photo).	Se dit navn og profilbillede
`email`	`email`, `email_verified`.	Se din e-mailadresse
`groups`	`groups`: `[{id, slug, name, role}]` for groups your app is installed in where the person is an active member. `role` is `member`, `moderator` or `admin`.	Se om du er medlem af de grupper, <app> er knyttet til
`verifications`	`verifications`: `{ <provider>: { verified_at, claims } }` for the person's live verifications.	Se dine verificeringer
`feed`	No claim. Lets your app read posts and comments as the person (section 5).	Vise dig opslag og kommentarer fra de grupper, <app> er knyttet til
`posts`	No claim. Lets your app post and comment in the person's name (section 5).	Skrive opslag og kommentarer i dit navn i grupper, du er medlem af

Ask only for what you use. Consent is remembered per app and accumulates; asking for a new scope later shows the consent screen again. People can remove access at any time in *Indstillinger → Forbundne apps*; from then on, App API calls on their behalf are refused.

4. The App API: authentication and conventions

The App API (`/api/v1/app/*`) is what your **server** calls. It authenticates with an **app token** from the client-credentials grant. A person's access token does not work here. You act on a person's behalf by naming their `sub`, and Theta checks their consent and membership.

```
curl -u 'bookingen:$CLIENT_SECRET' https://app.theta.dk/oidc/token \
  -d grant_type=client_credentials -d scope=app
# → { "access_token": "…", "token_type": "Bearer", "expires_in": 3600, "scope": "app" }

curl -H "Authorization: Bearer $APP_TOKEN" https://app.theta.dk/api/v1/app/groups
```

- **Bare JSON** bodies in and out. Ids are strings.
- Errors: `{ "error": { "code": "…", "message": "…", "fields": [ … ] } }`. An invalid body is `400` with one `fields` entry per problem.
- A group your app is not installed in, and every refused read, is `404`, so the API never reveals which people or groups exist.
- Writes are **idempotent** when you send an `externalId`: a retry returns `200` with the original instead of a duplicate.
- Rate limit: 300 requests per minute; a `429` carries `Retry-After`.

5. App API endpoints

Endpoint	Purpose
`GET /groups`	Groups the app is installed in: `{ groups: [{ id, slug, name, parentId, baseLocation }] }`.
`POST /posts`	Create a post, optionally with a card. `201 { id, url }`, or `200` on a repeat `externalId`.
`POST /posts/:id/comments`	Comment on a post, or reply when `:id` is a comment. `201 { id, url }`.
`GET /groups/:ref/posts?viewer=&cursor=&limit=`	Read a group's feed as a person (pinned first, then newest). `{ posts, nextCursor }`.
`GET /posts/:id/comments?viewer=&cursor=&limit=`	Read a post's thread as a person, depth-first. `{ comments, nextCursor }`.
`PUT /cards/:externalId`	Replace a card you posted (section 6). `{ id }`.
`POST /notifications`	Notify people (section 8). `{ delivered }`.

Authors: as the app, or as a person

- `"author": { "type": "app" }` writes as your bot account. Bot posts are trusted and not moderated.
- `"author": { "type": "user", "sub": "48213" }` writes in the person's name. It needs a live consent including `posts` and an active, unsuspended membership of the group; otherwise `403 userNotEligible` and nothing is written. These writes go through the group's normal moderation.
- Either way the post shows "via <app>".

```
POST /api/v1/app/posts
{
  "group": "solsikken-beboere",
  "author": { "type": "app" },
  "title": "Ledige weekender i fælleshuset",
  "body": "Book direkte i boksen herunder.",
  "externalId": "week-2026-49",
  "card": { …see section 6… }
}
```

Limits: `title` ≤ 200, `body` ≤ 20 000, `externalId` ≤ 200 characters. Polls are not allowed. Comments take `{ author, body, externalId? }`; an empty body is `422 bodyEmpty`.

Reading as a person

`viewer` must be a person with a live consent including `feed` who is an active member of the group. `limit` is 1–50 (default 20); a cursor belongs to one group or thread. Posts awaiting approval or hidden by moderators are never returned, so you can safely show one person's read to others.

```
{
  "id": "90211", "url": "https://app.theta.dk/opslag/90211",
  "author": { "sub": "48213", "name": "Mette Fiktiv", "picture": "https://…" },
  "title": "Fælles arbejdsdag", "text": "Vi mødes kl. 10 ved redskabsskuret.\n\nHusk handsker.",
  "createdAt": "2026-11-20T09:12:00.000Z", "editedAt": null,
  "commentCount": 4, "attachmentCount": 0, "viaApp": null
}
```

`text` is plain text (mentions as names, emoji as `:shortcode:`). Comments add `parentId` and `depth`. `author` is `null` for anonymous posts; `viaApp` is `{ name, postedAsApp }` for app-made content.

6. Cards (`theta.card/v1`)

A card is a small native UI block attached to a post, rendered by Theta on web and in the app. You send it with `POST /posts`, replace it with `PUT /cards/:externalId`, and return a new one when a button is pressed.

```
{
  "version": "theta.card/v1",
  "externalId": "faelleshus-2026-12-06",
  "title": "Fælleshuset, lørdag 6. dec.",
  "subtitle": "Plads til 40 · nøgle hos Hus 12",
  "badge": { "text": "Ledig", "tone": "success" },
  "fields": [ { "label": "Tid", "value": "10.00–23.00" }, { "label": "Pris", "value": "250 kr." } ],
  "actions": [
    { "id": "book", "label": "Book", "style": "primary", "confirm": "Book fælleshuset lørdag?" },
    { "id": "cancel", "label": "Aflys", "style": "danger", "audience": ["48213"] }
  ],
  "link": { "label": "Se kalender", "url": "https://booking.solsikken.example/kalender" }
}
```

Field	Rules
`title` / `subtitle` / `body`	≤ 120 / ≤ 160 / ≤ 500 characters; title required.
`badge.tone`	`neutral`, `info`, `success`, `warning`, `danger`.
`fields`	At most 6 `{ label, value }`.
`actions`	At most 3, unique `id`s. `style`: `primary`, `secondary`, `danger`. `audience`: list of `sub`s allowed to press (omit for everyone who can see the post). Optional `confirm` text.
`link.url`	Absolute https only.

Button presses

When someone presses a button, Theta sends a signed `dk.theta.card.action` event to your webhook URL and **waits up to 5 seconds** for the answer:

```
# event data
{ "cardExternalId": "faelleshus-2026-12-06", "actionId": "book", "postId": "90233",
  "user": { "sub": "48213", "name": "Mette Fiktiv" } }

# your answer: 200 with the new card (same externalId), optionally a message to the person
{ "card": { …badge "Booket af Mette"… }, "message": "Fælleshuset er booket til dig." }
```

Answer `400/403/404/409/422` with `{ "message": "…" }` to refuse; Theta shows your sentence and keeps the card. A timeout, a `5xx` or an invalid card shows "<app> svarede ikke. Prøv igen."

7. Webhooks: events from Theta

Every webhook is an HTTP `POST` of a **CloudEvents 1.0** structured-mode body (`application/cloudevents+json`), signed with your webhook secret:

```
X-Signature: t=1792584000,v1=<hex HMAC-SHA256(secret, "<t>.<raw body>")>

{ "specversion": "1.0", "id": "5f0c6b1e-…", "source": "https://app.theta.dk",
  "type": "dk.theta.group.posted", "subject": "groups/1029",
  "time": "2026-11-24T10:00:00.000Z", "datacontenttype": "application/json",
  "data": { "group": { "id": "1029", "slug": "solsikken-beboere" },
            "postId": "90240", "rootId": "90240", "url": "https://app.theta.dk/opslag/90240" } }
```

```
// Node.js: verify against the RAW body bytes, constant-time, ±300 s
import { createHmac, timingSafeEqual } from "node:crypto";
export function verify(secret, rawBody, header, now = Date.now() / 1000) {
  const { t, v1 } = Object.fromEntries(header.split(",").map((p) => p.split("=")));
  const mac = createHmac("sha256", secret).update(`${t}.${rawBody}`).digest("hex");
  const ok = v1?.length === mac.length && timingSafeEqual(Buffer.from(v1), Buffer.from(mac));
  return ok && Math.abs(now - Number(t)) <= 300;
}
```

- **Deduplicate on the event `id`.** Answer `2xx` within 5 seconds.
- On a non-2xx or timeout Theta retries after 10 s, 1 min, 5 min and 30 min (5 attempts in total), with the same `id` and a fresh signature each time.

Type	When, and `data`
`dk.theta.group.posted`	A post or comment becomes visible in an installed group (not your own writes). Ids only: `{ group, postId, rootId, url }`. Fetch content through the read endpoints.
`dk.theta.post.commented`	Someone comments on a thread your app started. `{ postId, externalId?, url, commentCount, comment: { id, authorName, excerpt, createdAt } }`; excerpt ≤ 140 chars, anonymous authors are "Anonym".
`dk.theta.group.member_joined` `dk.theta.group.member_left`	Membership changes in an installed group. `{ group: { id, slug }, user: { sub, name }, source: "manual" | "verification" }`.
`dk.theta.card.action`	A card button was pressed (section 6); answered synchronously.

8. Notifications

```
POST /api/v1/app/notifications
{ "recipients": ["48213", "50117"], "text": "Nøglen til fælleshuset ligger klar i Hus 12.",
  "href": "https://booking.solsikken.example/mine", "postId": "90233" }
# → { "delivered": 2 }
```

1–1 000 recipients; `text` ≤ 200 characters; `href` is an absolute https URL or a Theta `/path`. Only people who have consented to your app or are members of a group it is installed in are reached; others are skipped and not counted. Shown in the bell and as push as "<app>: <text>" with your logo, governed by the person's activity settings, never sent as e-mail.

9. Membership verification providers

For organisations with their own member system: here **Theta is the OIDC client** and your system is the provider. Theta runs an authorization-code flow with PKCE S256 and `client_secret_basic`, verifies your RS256 ID token against your JWKS (`iss`, `aud`, `exp`, `nonce`, and RFC 9207 `iss` when sent), and stores your claims as the person's verification.

- **You provide:** issuer with discovery, a client id and secret for Theta, the scopes to ask for, the labels (e.g. "Bekræft medlemskab", "Bliv medlem"), and optionally a claim to show as a badge.
- **You register** Theta's callback: `https://app.theta.dk/api/v1/me/verifications/<slug>/callback`. With "Bliv medlem", Theta adds `screen_hint=signup`.
- **Group rules** match your claims by dot path and JSON equality: `require` (needed to join) and `auto_join` (joined automatically while all rules hold). Rule-made memberships are removed when the claims stop matching; memberships made by admins or invitations are never touched.
- **Keep it current** by posting signed CloudEvents to `https://app.theta.dk/hooks/verification/<slug>` (same `X-Signature` scheme). `subject` is the person's id at your end; any type ending in `.created`, `.updated` or `.deleted` carries the full claims in `data`. Older events than the stored claims are ignored, so retries cannot undo a later change.

```
# Example rules for the fictional "Grønne Stier"
{ "group": "groenne-stier-forum", "provider": "groenne-stier",
  "kind": "require", "claimPath": "member", "equals": true }
{ "group": "groenne-stier-aarhus", "provider": "groenne-stier",
  "kind": "auto_join", "claimPath": "afdeling.kode", "equals": "aarhus" }
```

10. Limits and checklist

Item	Value
Post title / body / externalId	200 / 20 000 / 200 characters
Card title / subtitle / body / fields / actions	120 / 160 / 500 / 6 / 3
Notification text / recipients	200 characters / 1–1 000
Read page size	1–50, default 20
Webhook answer / card answer timeout	5 s / 5 s
Signature clock tolerance	±300 s
Rate limit	300 requests / minute

Before you go live: PKCE S256 on every login · secrets only in `Authorization: Basic` · verify ID tokens against JWKS · verify webhook signatures on the raw body · deduplicate events on `id` · send an `externalId` with every write · handle `403 userNotEligible` and `404` without retry loops · ask for the fewest scopes you need.

The examples in this document use fictional organisations, people and domains (`*.example`). Questions and registration: contact the Theta team.