

Forsyningsdigitaliseringsprogrammet

Tværgående Arkitektur og Udvikling (TAU) leverance 5 – Fælles retningslinjer for API for deling af forbrugsdata

Version	Ændringer	Dato
0.1	Udkast til review i arbejdsplan.	18-09-2026
0.2	Tilføjelser efter møde i arbejdsplan.	02-10-2026

Indholdsfortegnelse

Indledning og formål	2
Baggrund	2
Formål.....	3
Afgrænsning	6
Værdiskabelse	6
Arbejdsproces.....	6
Governance og forankring.....	6
Governance i regi af Forsyningsdigitaliseringsprogrammet.....	6
Forankring i sektoren.....	7
Retningslinjer for API.....	7
Overordnet teknologivalg.....	7
Metadata	7
Access tokens	8
RESTful principper og HTTP-metoder	9
Konsistens i struktur	9
JSON i svar	9
Opbygning af URL'er.....	10
Versionering	11
Politik for versionering	11
Udfasning af versioner.....	11
Versionsnummer i URL'er.....	12
Fejlhåndtering og fejlmeddelelser.....	12
HTTP-statuskoder	12
Detaljeret fejlmeddelelser	12
Dokumentation af API	13
Maskinlæsbar dokumentation	13

Non-funktionelle forhold	13
Tidsstandard	14
Paginerings.....	14
Datamodeller og dataschemaer	15
Ordliste	16
Referencer	18

Medlemmer af arbejdsgruppen:

- Steen Kramer, Chefkonsulent, Dansk Fjernvarme
- Martin Jensen, Sektionsleder, HOFOR
- Benjamin Mikkelsen, Solution Architect, Aarhus Vand
- Jeppe Steen Andersen, Dataarkitekt, Aarhus Vand
- Andreas Okkels Overgaard Thomsen, Senior Business developer, Center Denmark
- Peter Bruhn Andersen, IT-arkitekt, Digitaliseringsstyrelsen
- Peter West-Nielsen, Funktionsleder, Klimadatastyrelsen
- Morten Lindegaard, Chefkonsulent, Klimadatastyrelsen

FDP-sekretariatet

- Nicklas Vandfeldt Hansen, Specialkonsulent, Klimadatastyrelsen

Indledning og formål

Baggrund

Retningslinjerne for adgang og udveksling af forsyningsdata er udarbejdet af udvalget Tværgående Arkitektur og Udvikling (TAU) i regi af Forsyningsdigitaliseringsprogrammet (FDP). Disse retningslinjer tager afsæt i målsætning 2 om ensretning af data og digitalisering, som er fastsat af Forum for Forsyningsdata (FFD).

FDP har som mål at skabe et fælles grundlag for digitalisering i forsyningssektoren, og derfor er der igangsat et arbejdsforløb vedrørende interoperabilitet i regi af TAU. Arbejdsforløbet arbejder blandt andet med at udarbejde fælles retningslinjer for adgang og udveksling af forsyningsdata.

Retningslinjerne for adgang og udveksling af forsyningsdata udgør en del af TAU's leverance 5, "Fælles retningslinjer for adgang og udveksling af forsyningsdata", i arbejdsprogrammet for TAU under FDP.

Formålet med disse retningslinjer er at fastlægge en fælles standard for adgang til og udveksling af forsyningsdata, således at det bliver lettere at dele og bruge data på tværs af organisationer og systemer. Retningslinjerne skal danne grundlag for den videre udvikling af branchestandarder for adgang og udveksling af forsyningsdata.

Disse retningslinjer skal sikre, at forsyningsdata udveksles på en måde, der er sikker, effektiv og kompatibel med andre systemer og løsninger i forsyningssektoren, og at de kan håndtere de krav og udfordringer, der er forbundet med adgang til og udveksling af forsyningsdata.

Retningslinjerne henvender sig til IT-arkitekter, der udarbejder branchestandarder og domænespecifikke specifikationer eller designer IT-løsninger til deling af data med anvendere i forsyningssektoren, men det er tilstræbt, at retningslinjerne også kan læses af andre.

Flere oplysninger om Forsyningsdigitaliseringsprogrammet findes på:

www.forsyningsdigitaliseringsprogram.dk

Formål

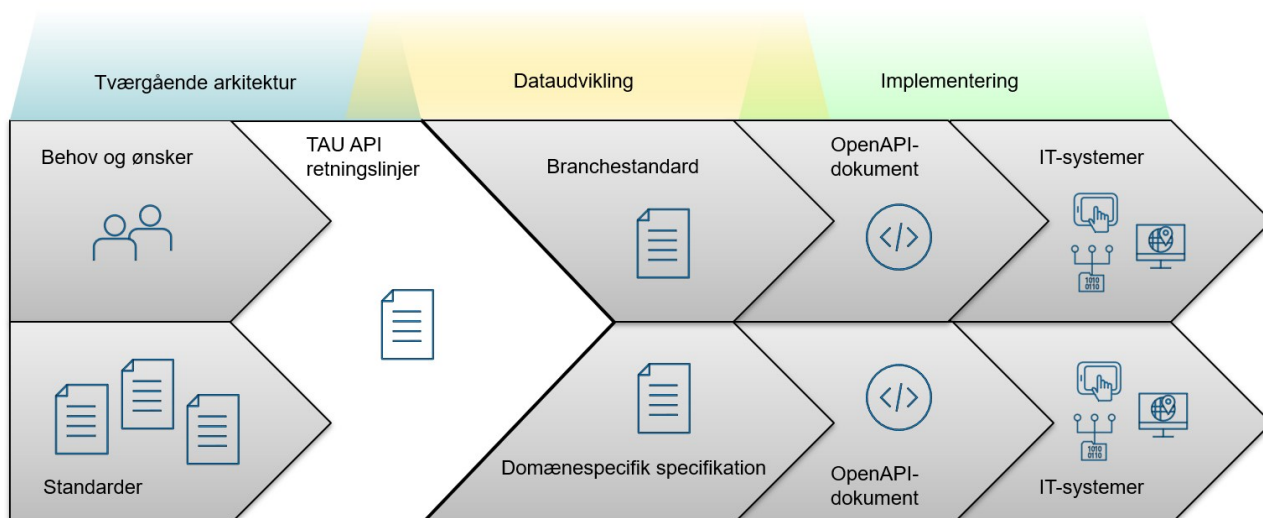
Dette dokument fastlægger de overordnede rammer for efterfølgende produkter fra dataudviklingsgrupperne (DUG'erne) med udgangspunkt i brugsscenarioet "historiske data". Det er ikke ment som en teknisk specifikation for konkrete IT-systemer, men som et fundament for at udarbejde branchestandarder, der kan danne grundlag for at implementere API-løsninger. Formålet er at sikre en ensartet og genanvendelig tilgang, der understøtter udviklingen af interoperable API'er og bidrager til en konsistent håndtering af historiske data på tværs af sektorer.

Brugsscenarioet "historiske data" fokuserer på udstilling af tidsseriedata, hvor eksterne anvendere henter (pull) historiske målinger fra forsyningselskaberne. Der er ingen funktionalitet for at notificere anvendere, hvis data bliver opdateret bagud i tid – kun læsning af indsamlede data som de er på det pågældende tidspunkt. Typiske use-cases er:

- **Se eget forbrug:** Privat- og erhvervsbrugere kan trække deres historiske forbrugsdata for at følge udviklingen over tid.
- **Forbrugstal til energirådgivning:** Energirådgivere anvender de historiske serier til at udarbejde analyser, identificere spidsbelastninger og foreslå energibesparende tiltag.
- **ESG-rapportering:** Virksomheder henter de nødvendige historiske energidata for at dokumentere deres CO₂-udledning og andre miljømæssige KPI'er i ESG-rapporter.

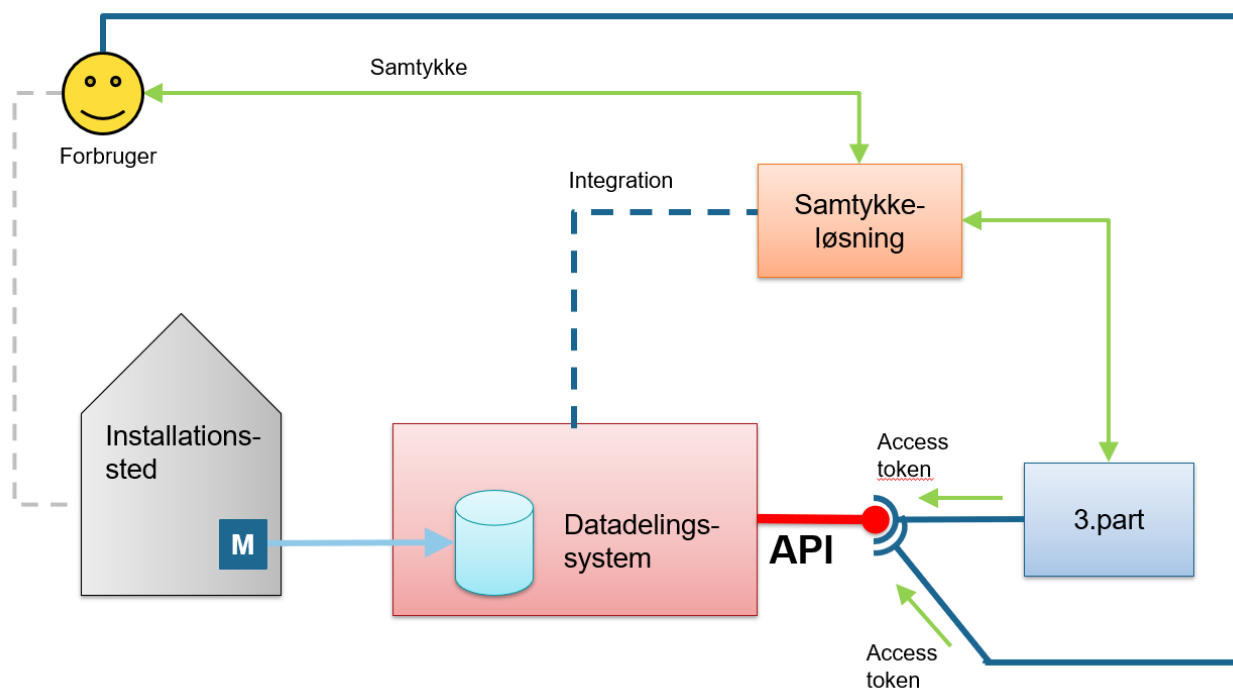
Alle data leveres via standardiserede API'er, hvilket sikrer ensartet adgang og let integration for tredjepartssystemer.

Dette dokument med fælles retningslinjer for API'er indgår i en proces, som er vist i nedenstående figur.



Figuren visualiserer processen hvor behov og ønsker og standarder danner grundlaget for fælles retningslinjer. Disse retningslinjer omsættes til branchestandarder eller lignende specifikationer, som igen udgør grundlaget for konkrete API-specifikationer. Til sidst realiseres specifikationerne i de faktiske IT-systemer, hvilket sikrer konsistens, genanvendelighed og interoperabilitet på tværs af sektorer.

Dette dokument omhandler API i et systemlandskab, der er visualiseret i nedenstående figur. Det pågældende API er fremhævet med rødt.



Figuren viser aktører og komponenter i scenarier for udveksling af forsyningsdata.

Et installationssted har målere. En måler har et eller flere måleregistre, der genererer måledata. Måledata overføres til et datadelingssystem, der udstiller et API. Et datadelingssystem er integreret med en samtykkeløsning af hensyn til adgangsstyring på API. Et installationssted er tilknyttet en eller flere forbrugere. En forbruger kan give samtykke til en tredjepart i samtykkeløsningen. Dette samtykke er forbrugerens viljestilkendegivelse om, at tredjepart må tilgå bestemt måledata fra et installationssted tilknyttet forbrugeren. Tredjepart kan fra samtykkeløsningen få access tokens, der kan benyttes til at tilgå data via API. Forbrugeren kan ligeledes få access tokens til at tilgå egne data via API.

API er her REST API, hvor måledata hentes med pull requests.

Et datadelingssystem kan være implementeret og drevet af en leverandør eller et forsyningsselskab. Overførsel af måledata fra målere til datadelingssystemer kan ske gennem andre systemer fra en leverandør. Disse overførsler kan ske via API, som kan være struktureret på samme måde som API udstillet af et datadelingssystem.

De termer, der optræder på figuren, er beskrevet i nedenstående tabel.

Term	Definition	Eksempler	Kommentar
------	------------	-----------	-----------

Forbruger	person, virksomhed el. lign. der bruger varer eller tjenesteydelser	Beboer i parcelhuset på Parcelvej 17 (person)	Baseret på FIBO (Consumer), jf. TAU begrebsmodel [1].
Installationssted	sted der forsynes og dermed et område, der dækker et internt forsyningsnet	Parcelhuset på Parcelvej 17 (fjernvarme)	Fra TAU begrebsmodel [1].
Måler	fysisk enhed der indeholder et eller flere målerregistre til at registrere en eller flere observationer for en eller flere egenskaber	Vandmåler 772 installeret i parcelhus på Parcelvej nr. 17 (vand)	Fra TAU begrebsmodel [1]. Illustreret med M i figuren.
Datadelingssystem	IT-system der udstiller API med forbrugsdata	Varmeoverblik.dk	
Samtykkeløsning	IT-system til administration af samtykker og tilhørende elektroniske artefakter		
Tredjepart	person eller virksomhed, der kan anvende andres forbrugsdata	Virksomhed, der rådgiver om energiforbedringer.	
Samtykke	en frivillig, informeret og eksplicit godkendelse fra en forbruger til at en tredjepart får adgang til den pågældende forbrugers forbrugsdata		Definitionen skal ses i kontekst af figuren, og udformningen af samtykket afhænger af udformningen af samtykkeløsningen.
Integration	udveksling af oplysninger mellem Samtykkeløsning og Datadelingssystem		Definitionen skal ses i kontekst af figuren, og udformningen af integrationen afhænger af udformningen af samtykkeløsningen.
API	et sæt af regler og protokoller der gør det muligt for software-komponenter at kommunikere for at udveksle data og udføre handlinger		Forkortelse af Application Programming Interface. Benyttes ofte synonymt med web-API, hvor kommunikationen foregår over internettet.
Access token	en tekststreng med oplysning der kan benyttes til identifikation og autorisere adgang		Konkret indhold og udformning af access tokens afhænger af den valgte standard for

			autorisering og implementering heraf.
--	--	--	---------------------------------------

Det bemærkes, at begrebsmodellen fra TAU leverance 2 [1] nævner delmodel om adgang til forbrugsdata med samtykke, tredjepart, dataejer mv. som videre arbejde. I den forbindelse kan nogle af begreberne i ovenstående tabel blive revideret.

Afgrænsning

Dette dokument omhandler kun overordnede rammer for API-design og branchestandarder. Det indeholder ikke specifikationer eller retningslinjer for en samtykkeløsning, og der er ingen anbefalinger vedrørende IT-sikkerhed medtaget. En samtykkeløsning behandles i en separat arbejdsrapport under TAU, og der henvises til gængse best practices og standarder for IT-sikkerhed.

Dokumentet fokuserer på API'er og forholder sig således ikke til løsninger med fildownload, som kan være relevante i situationer med meget store datamængder. Erfaringsmæssigt benyttes API'er med paginering dog også til fulde historiske udtræk af forbrugsdata.

Værdiskabelse

Standardiserede API'er giver ensartet adgang til data på tværs af forsyningsselskaber og -arter, hvilket gør det langt lettere for tredjepart at udvikle systemer og hente data. Genbrug af fælles komponenter og konfigurerbare opkoblinger erstatter unikt udviklede løsninger, så integrationerne bliver hurtigere og billigere. Samtidig vil en fælles ramme for samtykkestyring muliggøre en holdbar og enkel håndtering af samtykker fra forbrugere. Resultatet er ensartede branchestandarder og API'er, der er målrettet mod tredjeparter og dataanvendere, og som skaber øget effektivitet, innovation og interoperabilitet i hele forsyningssektoren.

Arbejdsproces

Udarbejdelsen af dokumentet startede med en afdækning i en konsulentrapport, som kortlagde nuværende behov, praksis og teknologiske muligheder. Afdækningen kom med konkrete anbefalinger til fælles retningslinjer for API til udveksling af forbrugsdata. På baggrund heraf nedsatte TAU en arbejdsgruppe bestående af repræsentanter fra forsyningssektoren samt organisationer og myndigheder med interesse i tværgående arkitektur. Gruppen afholdt en række workshops, hvor fokus var på, hvordan data kan udstilles til omverdenen, og på at foretage en teknologisammenligning (fordele/ulemper) for at vælge den mest egnede løsning. Resultatet er et sæt overordnede retningslinjer, der understøtter udarbejdelsen af branchestandarder og ensartet API-design i forsyningssektoren.

Governance og forankring

Dette dokument er ikke ment som et statisk artefakt, men skal kunne tilpasses en verden i konstant forandring. Forsyningssektorens behov udvikler sig løbende (fx nye forretningsmodeller, ændrede regulatoriske krav og teknologier), og derfor skal retningslinjerne for API-design kunne revideres. Der kan fx komme ændringer, hvis der etableres en central samtykkeløsning, og når standarder, bekendtgørelser og lovgivning opdateres. Der er behov for at etablere et aktivt governance setup med regelmæssige reviewcyklusser, klare ansvarsområder og en formaliseret ændringsproces for at sikre, at API-retningslinjerne forbliver relevante og i tråd med udviklingen i forsyningssektoren og omverdenen.

Governance i regi af Forsyningsdigitaliseringsprogrammet

I regi af FDP fungerer TAU som rådgivende organ, og kan iværksætte opdatering af de udarbejdede retningslinjer.

Forankring i sektoren

Brancheorganisationerne vil bidrage til at understøtte udbredelse og forankring af leverancen via udarbejdelse af branchestandarder, som formidles i deres respektive sektorer gennem egne kanaler, fx hjemmesider, nyhedsbreve og konferencer mv.

Retningslinjer for API

Retningslinjerne for API'er til deling af forbrugsdata er udformet som en række anbefalinger. I det følgende gennemgås først baggrunden for det overordnede teknologivalg. Herefter berører de efterfølgende afsnit en række forskellige emner med én eller flere anbefalinger for hvert emne. Hver anbefaling er ledsaget af introducerende og forklarende tekst samt rationale for anbefalingen og dens implikationer.

Overordnet teknologivalg

Det centrale i retningslinjerne er det overordnede valg af tilgang til API'er. REST API'er, også kendt som RESTful API'er, er i dag udbredte og betragtes som de-facto standarden for web-API'er. De er baseret på HTTP-protokollen og giver adgang til data via URL-adresser, idet HTTP-protokollens metoder (fx GET og POST) benyttes på en ensartet måde. Der er imidlertid også andre måder at tilgå data på. Mens REST API'er baserer sig på opbygningen af URL-adresser og parametre, så giver GraphQL adgang til data gennem et enkelt endpoint ved brug af et standardiseret sprog for forespørgsler. GraphQL er således et sprog (query-language), der standardiserer forespørgsler, og data udstillet i det specifikke endpoint er struktureret efter en datamodel, der er med til at dokumentere API'et. Styrken ved GraphQL er fleksibilitet i forespørgsler samt muligheder for at kombinere forskellige dataentiteter i svaret og udvælge specifikke felter i data. Mens man i REST API'er skal hente data og selv udvælge og kombinere dem, så kan passende GraphQL-forespørgsler foretage udvælgelse og kombinerende af data i generering af svaret. GraphQL benyttes fx på Datafordeleren, hvor man udnytter muligheder for at udvælge felter og kombinere data fra forskellige registre, og i Dataforsyningen, hvor både GraphQL og REST API'er bruges til at udstille tidsserier.

Forbrugsdata er tidsserier med få felter (fx dato, måling, måleenhed, målerId) og mange rækker. Der er derfor ikke et behov for at udvælge felter for at undgå at hente for mange data. Hvis datadelingssystemer udstiller data fra enkelte forsyningsselskaber, er der heller ikke umiddelbart et grundlag for at kombinere data i det enkelte API. De problemer, som GraphQL løser, har derfor ikke relevans for forbrugsdata i dag. Mht. samtykke og autorisation, så kan der i REST API'er foretages validering per ressource/endpoint, mens GraphQL kan være behov for en kontrol på feltniveau. Forsyningsbranchen benytter i dag REST API'er, og introduktion af GraphQL vil øge implementeringsomkostningen og forsinke udrulningen.

Ud fra betragtningerne omkring mønstret for forbrugsdata, håndtering af samtykker, modenhed i forsyningsbranchen samt drift og sikkerhed, så er REST API'er valgt som den overordnede tilgang. Hvis der senere etableres et centralt aggregeringslag på tværs af el, vand og varme, kan valg af teknologi genbesøges og brug af GraphQL overvejes igen.

I de efterfølgende afsnit behandles forskellige aspekter og detaljer omkring REST API'er og forhold omkring udstilling af data i API'er.

Metadata

Discovery metadata gør det muligt for anvendere at finde frem til datasæt udstillet med API på datadelingssystemet.



Anbefaling:

Udstil Discovery metadata som anvist i Fælles retningslinjer for forsyningssektorens metadata [2].

Der er allerede udarbejdet retningslinjer for metadata i forsyningssektoren, der tager højde for, at data ofte udstilles i et API. Det er derfor naturligt at følge disse retningslinjer frem for at etablere nye detaljerede anbefalinger.

Rationale: De data, som udstilles, skal være dokumenteret og deklareret i en sådan grad, at anvenderne kan fremsøge data, vurdere datas egnethed ift. et behov, få adgang til data, få indsigt i vilkår og evt. begrænsninger i at anvende data mm. (jf. [2]).

Implikationer: Discovery metadata skal oprettes, vedligeholdes og udstilles parallelt med driften af API med data i datadelingssystemet. I [2] anbefales, at man anvender udvalgte elementer fra standarden DCAT-AP-DK, og at man etablerer et centralt datasætkatalog til registrering, vedligeholdelse og udstilling af metadata. Indtil der ikke er etableret et centralt datasætkatalog, er det op til forsyningsselskaberne, hvorledes Discovery metadata gøres tilgængelige.

Access tokens

Nærværende retningslinjer for API vil dog ikke give detaljerede anbefalinger omkring access tokens og samtykke, men vil henvise til, at Forsyningsdigitaliseringsprogrammets øvrige arbejde med en eventuel central samtykkeløsning kan resultere i mere detaljerede anvisninger. Det anbefales dog, at man benytter access tokens, når man tilgår datadelingssystemets API som illustreret på systemlandskabet i et tidligere afsnit.



Anbefaling:

Benyt OAuth 2.0 Bearer-tokens som access tokens.

I OAuth 2.0 er Bearer-tokens den type access tokens, der er default¹. De er udformet som tekststreng, der sendes med i forespørgsler til API'et i en HTTP header (Authorization: Bearer).

Rationale: OAuth 2.0 er en velkendt standard, og brugen af Bearer-tokens er derfor kendt i forbindelse med web-API'er.

Implikationer: Enhver, der har et gyldigt token, kan få adgang til data, da der ikke kræves yderligere identifikation.

Det konkrete format for tokens er ikke specificeret i OAuth 2.0, men bestemmes af den konkrete implementering, dvs. her af den samtykkeløsning, der udsteder access tokens. Udover det konkrete format for tokens, skal der også fastlægges en politik for udstedelse af access tokens samt håndtering og invalidering af access tokens ud fra best practices.

¹ Se <https://oauth.net/2/bearer-tokens/> og <https://datatracker.ietf.org/doc/html/rfc6750> og <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-token-exchange-19>

RESTful principper og HTTP-metoder

REST API'er (også kendt som RESTful API'er) er HTTP-baserede web-API'er, der er vidt udbredt og understøttet på tværs af platforme og teknologier.

Et REST API giver adgang til data via specifikke URL-adresser i API'et, der repræsenterer data eller en handling. En række principper kan fremhæves:

- Alle URL'er skal kunne læses af mennesker og følge en basal struktur. Stier i API'ernes URL'er kan således bygges op efter en skabelon, som fx `{forretningsområde}/{versionsnummer}/{ressourcetype}/{ressourceid}`
- Der benyttes som udgangspunkt navneord i flertal for ressourcer, så fx *users* angiver brugere.
- Svaret fra et REST API er gerne en JSON-repræsentation af den ressource, der er udpeget med fx ressourcestype og -id i URL'en.

Ofte benyttes URL-parametre også for at fremsøge og filtrere data.



Anbefaling:

Udstil data i REST API'er over HTTPS.

Der er tale om en anbefaling af en overordnet tilgang. Andre anbefalinger kommer ind på flere detaljer for en række aspekter af REST API'er samtidig med, at branchestandarder og tilsvarende domænespecifikke specifikationer forventes yderligere at ensrette API'er.

Rationale: REST API'er i dag de-facto standarden for web-API'er, da det er enkelt at skalere og let at integrere med eksisterende systemer. Data tilgås via URL'er, der er lette at læse, når de er bygget systematisk op og har sigende parameternavne, fx kan man forestille sig at hente gennemsnitsværdier for måleren med nummer 123 for d. 1. august 2026 med 15 minutters intervaller med denne fiktive forespørgsel: <https://kds.dk/api/meters/123/readings?from=2026-08-01T00:00:00Z&to=2026-08-02T00:00:00Z&interval=15m&aggregation=avg>

I forespørgslen er det en del af stien i URL'en, at der er tale om måleværdier fra måleren med nummer 123, mens tidsintervallet for d. 1. august 2026 samt valg af gennemsnitsværdier med 15 minutters interval er givet ved parametrene from, to, interval og aggregation.

REST API'er er allerede i drift i forsyningsbranchen hos Eloverblik, flere værker og målerleverandører.

Implikationer: Datadelingssystemer skal implementere REST API'er med en struktur og evt. parametre, der fx kan være yderligere fastlagt i branchestandarder. Best practices for REST API'er mht. brug af HTTP-metoder skal følges, således at man implementerer et såkaldt "uniform interface" med metoderne GET, POST, PATCH, PUT og DELETE. Ved ren udstilling af data, vil et REST API dog typisk kun bruge GET og POST.

Konsistens i struktur

For at sikre nem og effektiv brug skal API'erne følge en ensartet struktur på tværs af alle end points, både med hensyn til strukturen på svar fra API'et og opbygningen af URL-adresser i API'et.

JSON i svar

JavaScript Object Notation (JSON) benyttes ofte i REST API'er.



Anbefaling:

Brug JSON som serialiseringsformat.

Hvis brugerne af API'et har meget forskellige behov, som ikke kan opfyldes af et enkelt serialiseringsformat, kan man understøtte Content Negotiation², hvor brugerne har mulighed for at vælge mellem flere serialiseringsformater, som passer til deres behov. Her skal alle formater dokumenteres og være ensartet på tværs af API'et. Den klare anbefaling er dog, at have et ensartet serialiseringsformat, så data altid udstilles og overføres på samme måde uanset den underlæggende datastruktur.

Rationale: Et konsekvent serialiseringsformat (her JSON) gør det lettere for klienter at parse og forstå data, reducerer fejl og understøtter bedre dokumentation og vedligeholdelse.

Implikationer: Når man implementerer et API, skal det som minimum udstille data i JSON-format. Det anbefales her at gøre brug af JSON, som er specificeret i RFC8259-standard³.

JSON er et af de dataformater, der anbefales i Fælles retningslinjer for dataformater for deling af forbrugsdata [3] til løsninger, hvor anvenderen henter data via et web-API. JSON som overordnet format beskriver ikke, hvordan data konkret er struktureret, og der bør være yderligere ensretning som beskrevet i afsnittet Datamodeller og dataschemaer.

Opbygning af URL'er

Det anbefales, at opbygningen af URL-adresser er ensartet. Det ligger uden for nærværende, overordnede retningslinjer at fastlægge egentlige API-specifikationer, men branchestandarder kan ensrette API'er ved at beskrive konkret opbygning af URL'er og forskellige parametre.



Anbefaling:

Fastlæg opbygningen af URL-adresser i API'er, herunder stier, parametre og URL-parametre i branchestandarder eller tilsvarende domænespecifikke specifikationer.



Anbefaling:

Byg API'er efter relevante branchestandarder eller tilsvarende domænespecifikke specifikationer.

Anbefalingen er to-delt, idet den både går på etablering af branchestandarder og tilsvarende specifikationer og deres efterfølgende brug.

Rationale: For let at genbruge softwarekomponenter til at tilgå forskellige API'er, bør URL-adresser hos forskellige datadelingssystemer være opbygget på samme måde. Det er imidlertid uden for scope af nærværende retningslinjer at beskrive en konkret opbygning af URL'er og forskellige parametre.

² https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Content_negotiation

³ <https://datatracker.ietf.org/doc/html/rfc8259>

Implikationer: Branchestandarder eller tilsvarende dokumenter skal indeholde detaljerede beskrivelser af API'er med opbygning af URL-adresser med endpoint og stier. Det skal angives, hvilke dele af stien, der fungerer som en parameter til at identificere en ressource. Parametre, der kan angives som URL-parametre i HTTP GET-forespørgsler eller som felter i en JSON-struktur i HTTP POST-forespørgsler, kan med fordel beskrives for at sikre en ensartethed på tværs af API'er implementeret i forskellige datadelingssystemer.

Versionering

Det er vigtigt at versionere API'er ved ændringer i funktionalitet eller data for at sikre klarhed og kompatibilitet.

Politik for versionering

Versionerede API'er ledsages af en politik omkring breaking changes (og non-breaking changes. Politikken skal være nedskrevet og tilgængelig for udviklere af API'et og anvendere af API'et.



Anbefaling:

Formulér en politik for versionering af API'et og gør denne politik tilgængelig for udviklere og anvendere af API'et.

Et eksempel på en breaking change er, at fjerne eller ændrer et felt i responset. Et eksempel på en non-breaking change er, at man godt må tilføje nye felter til responset.

Rationale: Denne politik guider udviklere til, hvornår versionsnummeret på et endpoint skal forøges. Derudover kan den hjælpe anvendere til at vide, hvilke ændringer, de skal kunne tage højde for.

Implikationer: Der kan være behov for at udstille politik for versionering på fx en landing page eller anden hjemmeside med information og vejledning om brug af API'et.

Udfasning af versioner

Når man versionerer sine API'er, er det ligeledes godt at have en deprecation-strategi, dvs. en strategi for udfasning af gamle versioner, samt en kommunikationsplan for at informere anvendere om planlagte ændringer.



Anbefaling:

Informér anvendere om strategi og planer for ændringer i API'et, herunder udfasning af versioner.

Rationale: Der er omkostninger forbundet med at have flere versioner af et API i drift, og der kan være datamæssige udfordringer. Derudover har anvendere behov for varsling og tid for at komme over på en nyere version af et API.

Implikationer: Der kan være behov for kommunikationskanaler, der løbende informerer anvendere. Perioder med flere versioner i (parallel)drift giver anvenderne tid til at skifte til nyere versioner og mindske generne hermed

Versionsnummer i URL'er

Der er flere forskellige måder, hvorpå man rent praktisk kan versionere REST API'er. En god standard er at bruge URL-baseret versionering, hvor man tilføjer versionsnummeret i URL'en. Det kan fx gøres sådan her: v1/users og v2/users



Anbefaling:

Lad versionsnummeret være en del af URL'erne i API'et.

Rationale: Det er klart, hvilken version af API'et, som forespørgslen er til.

Implikationer: Anvendere skal aktivt forholde sig til versioneringen.

Fejlhåndtering og fejlmeddelelser

Fejlmeddelelser skal være klare og informative, så anvendere og udviklere hurtigt kan identificere og rette problemer. Derudover bør fejlmeddelelser være konsistente på tværs af endpoints og API'er, hvorfor det er fordelagtigt med en standardiseret tilgang.

HTTP-statuskoder

Når REST API'er er baseret på HTTP-protokollen, er det naturligt at benytte de statuskoder, der i forvejen er defineret for HTTP, når der opstår fejlsituationer.



Anbefaling:

Benyt statuskoderne fra Hypertext Transfer Protocol (HTTP).

Oversigt og beskrivelser af statuskoder i HTTP kan findes her:

- <https://www.iana.org/assignments/http-status-codes>
- <https://datatracker.ietf.org/doc/html/rfc9110>

Rationale: Brug af velkendte statuskoder gør det lettere for anvendere at håndtere fejlsituationer.

Statuskoder benyttes til at indikere, om fejlen skyldes forhold på klientsiden, som anvenderen kan udbedre, eller om der er tale om en serverfejl.

Implikationer: Ved design og implementering af API'er skal brug af HTTP-statuskoder iagttages.

Detaljerede fejlmeddelelser

En anvender eller udvikler har lettest ved at forholde sig til en detaljeret fejlmeddelelse frem for blot en statuskode. Det er dog vigtigt, at fejlmeddelelser ikke afslører følsomme oplysninger. For at få en ensartet struktur på detaljerede fejlmeddelelser, kan man benytte sig af Problem Details som beskrevet i RFC 9457-standard⁴.

⁴ <https://datatracker.ietf.org/doc/html/rfc9457>



Anbefaling:

Benyt Problem Details til detaljerede fejlmeddelelser.

HTTP-statuskoder er ikke altid tilstrækkelige til at give den nødvendige information, og netop derfor anviser RFC 9457, hvordan mere detaljerede fejlmeddelelser kan gives i API'er ved hjælp af simple JSON-dokumenter.

Rationale: Man får en ensartet tilgang til detaljerede fejlmeddelelser ved at benytte en eksisterende, standardiseret tilgang. På den måde undgår man, at hvert datadelingssystem implementerer deres egen struktur på detaljerede fejlmeddelelser.

Implikationer: Ved design og implementering af API'er skal Problem Details tænkes ind, når der gives detaljerede fejlmeddelelser.

Dokumentation af API

Anvendere har behov for forskellige former for dokumentation af API'er.

Maskinlæsbar dokumentation

API'er skal dokumenteres, og REST API'er dokumenteres i dag ofte efter OpenAPI-specifikationen⁵. Selve opbygning af URL'er mv. i API kan fastlægges i branchestandarderne.



Anbefaling:

Dokumentér REST API'er ved brug af OpenAPI-specifikationen version 3.1 eller senere version.

Rationale: OpenAPI-specifikation af et API giver en maskinlæsbar beskrivelse af API'et, som kan benyttes af en række værktøjer i forbindelse med udvikling og test af integrationer.

Implikationer: Der udarbejdes en OpenAPI-specifikation i JSON eller YAML ved implementering af et API. Specifikationen kan udarbejdes som en del af API-designet, dvs. før implementeringen (contract first), eller som dokumentation. Specifikationer af API'er fra branchestandarder kan være udarbejdet i OpenAPI og fungere som skabeloner.

Non-funktionelle forhold

Udover de aspekter, der er givet ved elementer i en OpenAPI-dokument eller Discovery Metadata, kan der være behov for yderligere dokumentation.



Anbefaling:

Dokumentér non-funktionelle forhold omkring REST API'er for anvenderne.

⁵ <https://spec.openapis.org/oas/>

Rationale: Der kan være indført begrænsninger på anvenderes brug, fx rate limiting, som anvendere kan forholde konstruktivt til, hvis det er dokumenteret. Derudover kan det være nyttigt for anvendere at kende til fx supportfunktioners åbningstider, forventet opetid, servicevinduer og tidsrum med overvåget drift.

Implikationer: Der kan være behov for andre former for dokumentation end hvad der passer ind i OpenAPI-dokumenter og Discovery Metadata. Dette kan fx beskrives på en landing page for API'et.

Tidsstandard

Når man arbejder med tidsseriedata er det vigtigt at holde et ensartet og konsistent dato- og tidsformat.



Anbefaling:

Brug ISO 8601 ved angivelse af datoer og tid.

ISO 8601 standarden⁶ er en velkendt, udbredt standard.

Rationale: Det er vigtigt at holde et ensartet og konsistent dato- og tidsformat for at sikre, at der ikke er usikkerheder i forhold til, hvilken dato eller tid der menes.

Implikationer: Benyt en teknologistak, der i forvejen implementerer ISO 8601.

Bemærk at tid kan angives som lokal tid, som UTC (Coordinated Universal Time) eller med en tidszone (offset fra UTC):

- Dato og tid skrevet som "2026-10-01T16:35:00.000" fortolkes som lokal tid, da der ikke er angivet information om tidszonen.
- Dato og tid skrevet som "2026-10-01T14:35:00.000Z" er i UTC-tid, angivet med suffix "Z".
- Dato og tid skrevet som "2026-10-01T16:35:00.000+02:00" er dansk sommertid. Dansk normaltid eller vintertid angives med suffix "+01:00".

Erfaringsmæssigt kan brug af lokal tid give anledning til uklarheder ved overførsel af data mellem it-systemer, og det kan derfor være en fordel at angive UTC eller eksplicit tidszone.

Paginering

API'er bør understøtte paginering for at håndtere større datamængder. Paginering bør bruges alle steder, hvor man ikke kun spørger efter en enkelt entitet, men om en samling af entiteter.



Anbefaling:

Understøt paginering, når svaret kan være en liste af elementer.

Der er forskellige tilgang til paginering. For at få ensartet tilgang til paginering på tværs af API'er, kan man benytte de samme parametre i de tilfælde, hvor man vurderer, at de er tilstrækkelige i forhold til datamængder og forventet anvendelse af API'et.

⁶ <https://www.iso.org/iso-8601-date-and-time-format.html>



Anbefaling:

Benyt URL-parametrene offset og limit, hvor en simpel paginering er tilstrækkelig.

Parameteren *offset* angiver antallet af elementer, der springes over i svaret. Dvs. hvis en forespørgsel som udgangspunkt fremsøger en liste med 142 elementer, så vil offset=30 bevirke, at der returneres en liste uden de første 30 elementer.

Parameteren *limit* angiver det maksimale antal elementer, der skal returneres.

Man kan kombinere offset og limit for at hente et større antal elementer lidt ad gangen.

Defaultværdier og API'ets virkemåde for limit=0 bør dokumenteres over for anvenderen, da der er forskellige kendte tilgange til dette. I nogle API'er benyttes limit=0 til at give bedømt om alle elementer og dermed deaktivere en defaultværdi for limit, mens andre API'er returnerer nul elementer.

Rationale: Der behov for paginering for at kunne begrænse data i svar fra API'et. Hvis det er muligt at benytte ens parametre i forskellige API'er, øger det mulighederne for genbrug af komponenter.

Implikationer: Datadelingssystemer skal indeholde funktionalitet til paginering, og anvendere skal kunne håndtere den form for paginering, som det relevante datadelingssystem tilbyder.

Datamodeller og dataschemaer

Branchestandarder for API bør indeholde veldefinerede datamodeller og tilhørende dataschemaer for de data, der udveksles. Uden fælles datamodeller risikerer forsyningselskaber og leverandører at fortolke de samme begreber forskelligt, hvilket underminerer den interoperabilitet, som disse retningslinjer skal understøtte.



Anbefaling:

Specificér entydige datamodeller og maskinlæsbare dataschemaer (JSON Schema) for de datastrukturer, der udstilles via API'et, i branchestandarder og tilsvarende domænespecifikke specifikationer.



Anbefaling:

Benyt datamodeller og dataschemaer (JSON Schema) fra relevante branchestandarder eller tilsvarende domænespecifikke specifikationer.

En datamodel beskriver de begreber, attributter og relationer, som API'et arbejder med (fx måling, tidsstempel, enhed og kvalitetsflag), mens et dataschema formaliserer denne model i et maskinlæsbart format, der kan bruges til automatisk validering af requests og responses. Sammen sikrer de, at:

- **Entydig integration:** Tredjeparter kan udvikle integrationer ud fra en præcis og entydig specifikation, uden at skulle basere sig på eksempler eller mundtlig viden.
- **Automatisk validering:** Data kan valideres automatisk mod schemaet, både hos afsender og modtager, hvilket reducerer fejl og understøtter en ensartet kvalitet på tværs af forsyningsarter.

- **Styret versionering:** Ændringer i datamodellen kan håndteres systematisk gennem versionering af schemaet, så bagudkompatibilitet og en klar migreringsvej sikres for eksisterende integrationer.
- **Konsistent dokumentation:** Dokumentationen af API'et (med OpenAPI) kan genereres eller valideres direkte ud fra schemaet, så dokumentation og faktisk implementering ikke går ud af trit med hinanden.

Det er i branchestandarderne, at de konkrete datamodeller udarbejdes og vedligeholdes, men det bør fremgå som et krav i disse fælles retningslinjer, at enhver branchestandard skal indeholde en formel datamodel og et tilhørende schema som en integreret del af standarden – ikke som et efterfølgende eller valgfrit tillæg.

Rationale: Uden dette krav er der risiko for, at der bliver udviklet branchestandarder med forskellig detaljeringsgrad og struktur for datamodellerne, hvilket vil svække den ensartethed og genanvendelighed, som er dokumentets overordnede formål.

Implikationer: Kræver, at der stilles krav til, hvilket schemaformat der skal anvendes (JSON Schema, jf. valget af JSON som serialiseringsformat), samt en proces for versionering og governance af datamodellerne i takt med at branchestandarderne udvikler sig.

Ordliste

API: Application Programming Interface. Et API er en snitflade til software, der kan benyttes ved udvikling af softwareapplikationer. I dag benyttes API ofte synonymt med web-API, således at det betegner en snitflade, der kan tilgås over internettet, fx med HTTP-protokollen.

Datafordeleren: Egentlig "Den Fællesoffentlige Datafordeler". Datafordeleren er distributionsplatform for grunddata som fx Danmarks Adresser, data om personer (CPR), virksomheder (CVR) samt bygninger og boliger (BBR).

Dataforsyningen: Distributionsplatform, der giver adgang til frie offentlige geografiske data via kortvisning, webservices, webapplikationer, API'er og download.

DCAT-AP: Står for Data Catalog Vocabulary - Application Profile. Det er en europæisk profil af DCAT, som er en W3C-standard til metadata om datasæt.

DCAT-AP-DK: Bygger videre på den europæiske DCAT-AP-standard, men tilføjer danske termer, klassifikationer og retningslinjer.

Discovery metadata: Refererer til metadata, der hjælper med at identificere og finde datasæt, som er relevante for et bestemt behov eller interesse. Det giver vigtig information, der gør det muligt for brugere at opdage og forstå, hvilke data der er tilgængelige, hvor de er opbevaret, og hvordan de kan tilgås. Kort sagt besvarer discovery metadata spørgsmålet: "Hvilke datasæt indeholder den type data, jeg er interesseret i?" (Se også Metadata.)

endpoint: Termen benyttes ofte i betydningen URL, der er en digital adresse, men et endpoint dækker også over kombinationen af en URL og en specifik HTTP-metode (fx GET eller POST), således at der er tale om en specifik funktion eller ressource.

ESG-rapportering: ESG står for Environmental, Social and Governance, og EU-direktivet Corporate Sustainability Reporting Directive (CSRD) stiller krav om virksomheders bæredygtighedsrapportering med ESG-data.

GraphQL: GraphQL er et sprog til forespørgsler (et query language) for API'er. Det tilbyder stor fleksibilitet med forespørgsler, der kan kombinere relaterede data og udvælge præcis de felter i data, som anvendere har brug for.

HTTP: Står for Hypertext Transfer Protocol. Det er en protokol, der bruges til at udveksle informationer over internettet mellem klienter og servere.

ISO: ISO er det korte navn for International Organization for Standardization. ISO er en uafhængig international organisation, der udgiver en lang række internationale standarder. Man refererer typisk til standarderne et nummer. Fx er ISO 8601 den korte betegnelse for Date and time format, som i dag er opdelt i flere dele, ISO 8601-1:2019 og ISO 8601-2:2019.

JavaScript: Programmeringssprog designet til internettet. JavaScript kører i webbrowserne og benyttes til at implementere funktionalitet på hjemmesider.

JSON: JSON (JavaScript Object Notation) er et tekstbaseret, menneskeligt læsbart dataudvekslingsformat. Der er en bred understøttelse med forskellige programbiblioteker, og som navnet antyder, er det baseret på JavaScript-syntaks. JSON benyttes i dag i mange API'er og webudvikling.

KPI: Står for Key Performance Indicator (nøgletal), der er en målbar værdi, der bruges til at vurdere, hvor effektivt en organisation, afdeling eller proces når sine strategiske mål.

Metadata: For at kunne understøtte anvendelsen af data er det nødvendigt med oplysninger om, hvordan man har indsamlet data, hvad et datasæt i udgangspunkt kan anvendes til, hvem der har ansvaret for datasættet, og hvad kvaliteten af det er. Det hedder under ét metadata og er en slags varedeklaration.

OpenAPI: Et format og sprog til at beskrive snitflader til HTTP RESTful API'er på en standardiseret måde. Både JSON og YAML kan benyttes, og OpenAPI-beskrivelser er både menneske- og maskinlæsbare. OpenAPI startede som Swagger, og der findes derfor værktøjer under navnet Swagger, der understøtter OpenAPI.

Pull: Mønster for udveksling af data, hvor anvenderen initierer hentning af data. (I modsætning til "push", hvor data sendes til anvenderen, når der er nye eller ændrede data.)

REST: Står for REpresentational State Transfer. REST er en stil inden for softwarearkitektur, der er udbredt for web-baserede systemer.

RFC: Står for Request for Comments. RFC'er er tekniske dokumenter udgivet af Internet Engineering Task Force (IETF), der beskriver forskellige aspekter af internettet.

URL: Står for Uniform Resource Locator. En URL er en digital adresse på en ressource på internettet, fx en hjemmeside eller en fil.

YAML: Står for "YAML Ain't Markup Language". Det er et menneskelæsbart serialiseringsformat, som ofte benyttes til konfigurationsfiler, da det er designet til at være let at læse for mennesker, samtidig med at det stadig er struktureret til at være maskinlæsbart.

W3C: Står for The World Wide Web Consortium. W3C udvikler standarder og retningslinjer for web-baserede teknologier for at fremme principper om tilgængelighed, internationalisering, privatliv og sikkerhed.

Referencer

- [1] TAU-arbejdsspor 2 vedr. interoperabilitet, »Leverance 2: Fælles retningslinjer for begreber og deres relationer,« 11 april 2025. [Online]. Available: <https://www.forsyningsdigitaliseringsprogram.dk/produktbibliotek/tau-produkter>.
- [2] TAU-arbejdsspor 2 vedr. interoperabilitet, »Leverance 4: Fælles retningslinjer for forsyningssektorens metadata,« 28 juli 2025. [Online]. Available: <https://www.forsyningsdigitaliseringsprogram.dk/produktbibliotek/tau-produkter>.
- [3] TAU-arbejdsspor 2 vedr. interoperabilitet, »Leverance 3: Fælles retningslinjer for dataformater for deling af forbrugsdata,« 19 august 2025. [Online]. Available: <https://www.forsyningsdigitaliseringsprogram.dk/produktbibliotek/tau-produkter>.