ADDML 8.3

Archival Data Description Markup Language, versjon 8.3

Generell del

Versjon 1.0
Sist oppdatert: 2019-11-06 TPD

Innledning

Dette dokumentet beskriver det norske arkivverkets standard for teknisk metadata - Archival Data Description Markup Language (ADDML) versjon 8.3.

Dokumentet består av følgende tre hoveddeler

- en innledende del med overordnede opplysninger
- en del som beskriver selve ADDML og de muligheter som ligger i standarden
- en del som beskriver Arkivverkets bruk av standarden.

ADDML er en standard for å beskrive samlinger av datafiler. En slik samling kalles et datasett og en fil som inneholder beskrivelsen av datasettet, kalles en datasettbeskrivelse.

Mål

ADDML er en standard for å beskrive samlinger av datafiler som er organisert som flate filer. (En flat fil i denne sammenheng betyr at filen er i ren tekst og internt organisert enten ved fast posisjonering eller ved tegnseparasjon.)

En slik samling av filer kalles et datasett.

En fil som inneholder beskrivelsen av datasettet kalles en datasettbeskrivelse.

Historie

ADDML har hatt tidligere versjoner, men kun et par av disse har vært i reell bruk. Nemlig versjon 7.2 og 7.3. Disse ble benyttet ved utviklingen av Arkadukt 1.0 og Arkade 1.0. Tidligere versjoner av ADDML ble utviklet samtidig med utviklingen av programvaren og ble følgelig aldri tatt i bruk.

Når programmene var av en slik karakter at de kunne testes ut var standarden ADDML allerede oppe i 7.2. Denne ble benyttet en stund, men fortsatt ble det avdekket mangler, slik at det etter en stund ble laget en ny versjon som fikk nummeret 7.3. Denne versjonen ble så benyttet i en del år på begynnelsen av 2000-tallet.

Ved utviklingen av ny versjon av programvaren Arkadukt og Arkade ble det også tatt beslutninger om å forenkle ADDML-standarden. Dette ble så gjort i et par runder og medførte så omfattende endringer at versjonsnummeret ble endret til 8. De første versjonene (8.0 og 8.1) ble igjen bare mellomversjoner inntil man kom frem til en stabil versjon med 8.2. En ny versjon (8.3) vil kom ved utgangen av 2014. Denne versjonen inneholder kun mindre justeringer sett i forhold til 8.2.

Med versjon 8.2 kom også et nytt regime for vedlikehold av standarden. Forut for denne versjonen ble det underskrevet en avtale mellom riksarkivarene i Norge og Sverige om en felles utvikling siden begge landenes riksarkiv benyttet standarden. Siden er også Finland kommet med i samarbeidet.

De første versjonene av standarden var beregnet på å beskrive strukturen for flate filer. I versjon 7.x ble det også lagt inn en mulighet for å beskrive xml-filer. I versjon 8.x er dette tatt ut igjen slik at struktur igjen kun fortsatt gjelder for flate filer. Å detalj-beskrive en xml-fil er det ikke sett noen hensikt med siden man normalt vil ha en slik beskrivelse i form av et xml-skjema (.xsd).

Hvordan benytte ADDML

ADDML kan benyttes til mange formål. Den opprinnelige tanken bak ADDML var for å beskrive den tekniske strukturen til datasett som avleveres eller deponeres til et depot. I dag er standarden utvidet i forhold til sitt opprinnelige formål.

Den tekniske strukturen er fortsatt med slik at man kan beskrive strukturen for flate filer når disse skal overføres fra et system til et annet.

I versjon 8.x er det også mulig å beskrive andre typer filer, dog uten å gå i detalj. Det er mer lagt vekt på å kunne beskrive typen av filer, sammenhengen disse imellom, osv.

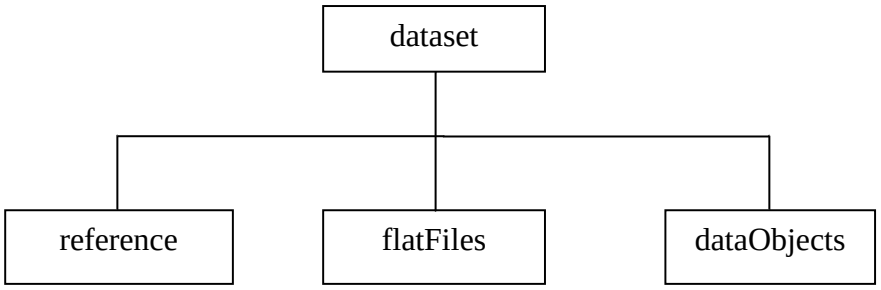
Både delen *reference* og delen *dataObjects* er generisk slik at man kan bygge ut etter eget behov. Dessuten er det også lagt inn muligheten for egenskaper som også kan benyttes på samme måte.

Når man tar i bruk ADDML er det derfor viktig at man selv setter begrensninger. Man må selv definere hvordan man benytter de generiske delene av standarden. Dette anbefales man å gjøre ved å lage en egen profil som beskriver ens egen bruk av standarden.

Beskrivelse av ADDML

Hva beskrives i ADDML 8.3. (Hovedstruktur)

ADDML består av 3 hoveddeler. En del som beskriver tilhørende informasjon (kataloginformasjon), dvs. informasjon om dataene i datasettet - *reference*. En del som er selve beskrivelsen av flate filer i detalj - *flatFiles*. En del som gir mulighet for å knytte opp andre filer enn flate filer til beskrivelsen - *dataObjects*.



Figur 1. Hoveddelene i ADDML.

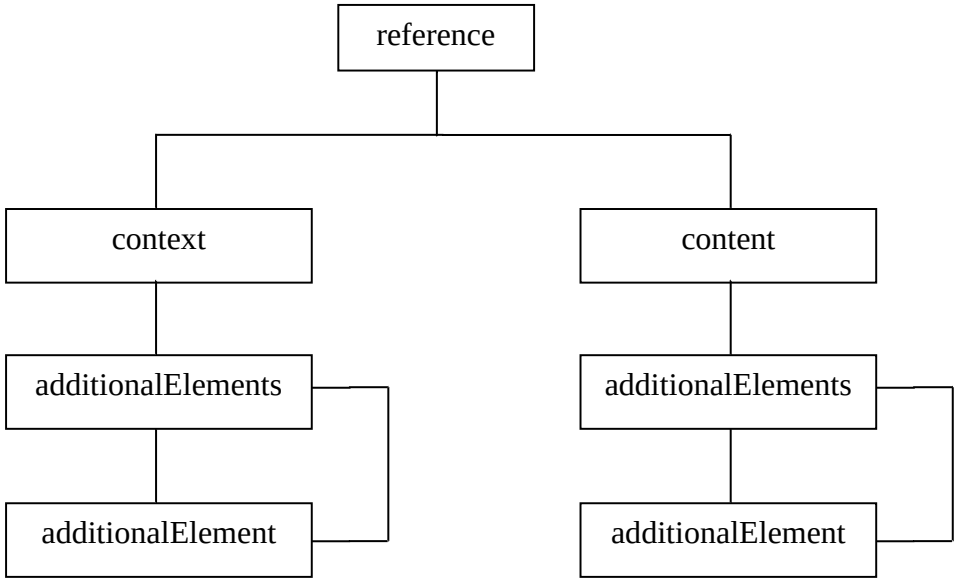
I *reference* vil det kunne registreres bevaringsverdige metadata på kontekstuell og innholdsmessig nivå. I *flatFiles* vil man kunne registrere en detaljert beskrivelse av strukturen i datafilene dersom de enten er i fast format eller i tegnseparert format. Alle filer som ikke er i fast format eller tegnseparert format vil bli beskrevet i *dataObjects*. Dog vil det her ikke være mulig å gi en detaljert beskrivelse.

I en addml-fil vil disse toppnivåene se ut som følger:

*<addml name=”***addmlnavn***”>*
    *<dataset name=”***datasettnavn***”>*
        *<reference name=”***refnavn***”>*
            …..
        </reference>
        <flatFiles>
            .....
        </flatFiles>
        <dataObjects>
            .....
        </dataObjects>
    </dataset>
</addml>

Merk at samtlige hoveddeler ikke er obligatoriske i seg selv, slik at en tom addml-fil er tillatt. Man kunne endret standarden ved å kreve minst en av hoveddelene, men en tom addml-fil er ansett som unaturlig. Tidligere var *reference* obligatorisk, men dette er en av endringene fra 8.2 til 8.3.

Tilhørende informasjon – bevaringsverdige metadata.



Figur 2. Oversikt over elementene i *reference*.

Med tilhørende informasjon menes informasjon om datasettet. Dette kan være opplysninger om hvem som har laget datasettet, hva slags type system det er hentet fra, bakgrunn for dataene i datasettet, osv. Dette er IKKE beskrivende informasjon av selve datasettet.

I ADDML er all slik informasjon samlet i *reference*. *reference* har to underelementer. Disse beskriver følgende:

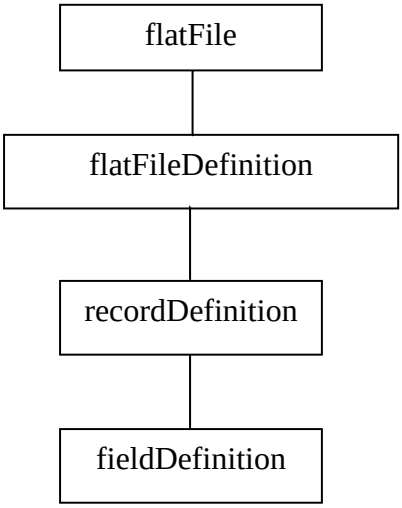
- kontekstuell informasjon - *context*
- innholdsrelatert informasjon - *content*

Grupperingene har ingen faste felter, men det kan defineres egne felter som kan eller skal medfølge et datasett. Se eget avsnitt om generiske elementer.

Et eksempel på *reference* med underliggende elementer kan da se ut som følger:

```
<reference>
  <context>
    <additionalElements>
      <additionalElement name="agents">
        <additionalElements>
          <additionalElement name="agent">
            <additionalElements>
              <additionalElement name="recordCreator">
                <value>Riksarkivet</value>
              </additionalElement>
            </additionalElements>
          </additionalElement>
        </additionalElements>
      </additionalElement>
    </additionalElements>
  </context>
  <content>
    <additionalElements>
        <additionalElement name="archivalPeriod">
            <additionalElements>
                <additionalElement name="startDate">
                    <value>20040401</value>
                </additionalElement>
                <additionalElement name="endDate">
                    <value>20100331</value>
                </additionalElement>
                <additionalElement name="period">
                    <additionalElements >
                        <additionalElement name="inngåendeSkille">
                            <value>Skarpt</value>
                        </additionalElement>
                        <additionalElement name="utgåendeSkille">
                            <value>Mykt</value>
                        </additionalElement>
                    </additionalElements>
                </additionalElement>
            </additionalElements>
        </additionalElement>
    </additionalElements>
  </content>
</reference>
```

Strukturen for å beskrive flate filer.



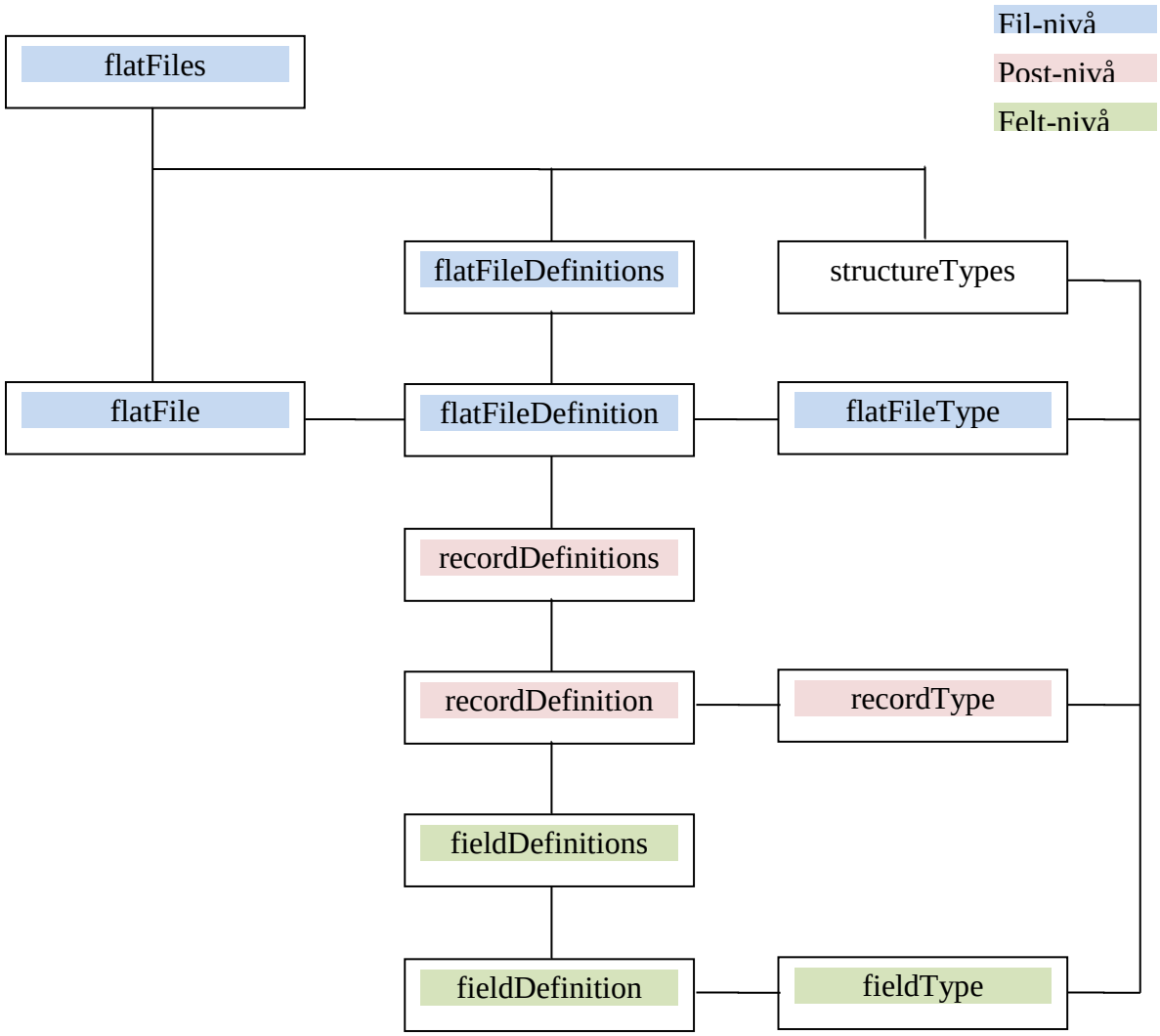
Figur 3. Oversikt over elementene i *dataset* (forenklet form).

Struktur delen i ADDML er en struktur for å beskrive filer som er av typen flate filer. Med flate filer menes at filen enten er med fast format (tabellform) – hvor alle felt starter i samme posisjon – eller tegnseparert format – hvor feltene er adskilt med et nærmere angitt tegn som skille mellom feltene (csv-filer er f.eks. på denne formen).

Figur 3 viser modellen i en forenklet versjon. Strukturen har et *flatFile* som øverste nivå. En *flatFile* vil inneholde *flatFileDefinition* , som igjen vil inneholde en eller flere *recordDefinition*. Og deretter vil så *recordDefinition* inneholde en eller flere *fieldDefinition*.

I en vanlig relasjonell database vil det normalt ikke være behov for post-nivået. Hvilket betyr at fil-nivået og post-nivået har smeltet sammen til ett og samme nivå – en tabell i en slik database. I andre sammenhenger kan imidlertid en fil inneholde mange forskjellige typer av poster som man ønsker å splitte opp til hver sin tabell. Koblingen mot en database vil derfor være mot post-nivået og ikke mot fil-nivået. Av denne grunnen er derfor også alle informasjoner om nøkler lagt på post-nivået.

Den komplette modellen inneholder tre hoveddeler. Dessuten består hvert nivå av et multiplums-nivå og et detalj-nivå, med unntak av *flatFiles*. De tre delene inneholder gjennomgående informasjoner om den fysiske representasjonen av nivået, den definisjonsmessige og en overordnet typedefinisjon. Det er imidlertid i dag ikke funnet behov for den fysiske representasjonen på annet enn fil-nivå. Skulle et slikt behov melde seg senere vil det være behov for en revisjon av standarden.



Figur 4. Oversikt over elementene i *flatFiles* (komplett form).

Dette betyr at strukturen er blitt mer kompleks enn i tidligere versjoner, men samtidig mer fleksibel og generell for bruk.

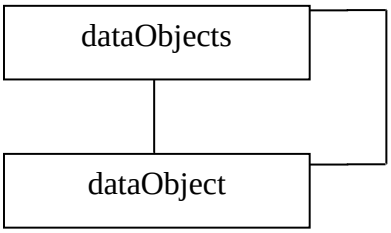
De øverste nivåene i flatFiles vil da kunne se ut for eksempel som dette:

```
<flatFiles>
    <flatFile name="filnavn" definitionReference="fildef1">
    </flatFile>
    <flatFileDefinitions>
        <flatFileDefinition name="fildef1" typeReference="typefildef1">
            <recordDefinitions>
                <recordDefinition name="postdef1" typeReference="typepostdef1">
                    ....
                    <fieldDefinitions>
                        <fieldDefinition name="fodselnr" typeReference="typefeltdef1">
                            ....
                        </fieldDefinition>
                    </fieldDefinitions>
                </recordDefinition>
            </recordDefinitions>
        </flatFileDefinition>
    </flatFileDefinitions>
    <structureTypes>
        <flatFileTypes>
            <flatFileType name="typefildef1">
                ....
            </flatFileType>
        </flatFileTypes>
        <recordTypes>
            <recordType name="typepostdef1"/>
        </recordTypes>
        <fieldTypes>
            <fieldType name="typefeltdef1">
                ....
            </fieldType>
        </fieldTypes>
    </structureTypes>
</flatFiles>
```

I eksemplet er det med farger påvist hvordan referansene går mellom det fysiske nivået til definisjonsnivået for fil ved hjelp av ***fildef1***. Dessuten tilsvarende mellom definisjonsnivåene og typenivåene for fil ved hjelp av ***typefildef1***, for post ved hjelp av ***typepostdef1*** og for felt ved hjelp av ***typefeltdef1***.

Andre filer enn flate filer.

Som nevnt over inneholder strukturen kun definering av flate filer med fast eller tegnseparert format. Andre filtyper kan ikke beskrives i detalj vha. ADDML 8.3-elementer. Dog er det mulig å knytte andre typer filer og eller informasjonsobjekter opp mot datasettet som defineres i en ADDML-fil. Dette gjøres ved å definere disse filene/informasjonsobjektene som logiske objekter ved å bruke elementene *dataObjects* og *dataObject*. På et logisk objekt kan det så knyttes opp en del egenskaper for å forklare hva slags filer/informasjonsobjekter dette er. (Filene kan være datafiler i xml-format, dtd eller xml-skjema, dokumentfiler, bildefiler, lydfiler, videofiler, osv.)



Figur 5. Oversikt over elementene i *dataObjects*.

Følgende kan være et eksempel på bruken av dataObjects:

<dataObjects>
*<dataObject name=”***Rapporter***”>*
<dataObjects>
*<dataObject name=”***rapportfil***”>*
….
</dataObject>
</dataObjects>
</dataObject>
</dataObjects>

Generiske elementer.

I de forskjellige beskrivelsene over har det dukket opp flere steder elementer som danner en loop. Det gjelder både for *additionalElements – additionalElement* og *dataObjects – dataObject*. I disse tilfellene er det snakk om en generisk struktur hvor den som benytter standarden selv kan bygge opp en hierarkisk struktur med de nevnte elementene.

Egenskaper.

For mange elementer er det i tillegg muligheten til å legge på egenskaper i form av attributtet *properties*. Også egenskaper har som tilleggselementer en generisk struktur ved *properties – property*.

Herunder vises et eksempel på bruk av egenskaper:

```
<dataObject name="rapportfil">
    <properties>
        <property name="filnavn">
            <value>rapport.xml</value>
        </property>
        <property name="sjekksum">
            <properties>
                <property name="algoritme">
                    <value>SHA-256</value>
                </property>
                <property name="verdi">
                    <value>
F13CED809E4AD36198352495397FABB54DCECCBD5A33BEEDB50BBDD5C9A09232
</value>
                </property>
            </properties>
        </property>
    </properties>
</dataObject>
```

Prosesser.

Standarden gir i tillegg brukerne muligheten til å definere egne operasjoner som skal utføres på informasjonen. Dette gjøres ved bruk av elementet *processes* som kan inneholde et sett av elementene *process*, som definerer den enkelte operasjonen. Eksempler på operasjoner som kan tenkes er kontroller (f.eks. kontrollere sjekksum, koder), konverteringer (f.eks. pakke opp pakkede felt, endre fra EBCDIC til ASCII eller UTF-8), osv.

Kontroll av koder for elementet yrke i posttypen postdef1 i filen fildef1 kan f.eks. se slik ut:

```
<flatFileProcesses flatFileReference="fildef1">
  <recordProcesses definitionReference="postdef1">
    <fieldProcesses definitionReference="yrke">
      <processes>
        <process name="Control_Codes"/>
      </processes>
    </fieldProcesses>
  </recordProcesses>
</flatFileProcesses>
```

Når det gjelder generiske elementer, egenskaper og prosesser er det helt opp til brukerne av standarden å definere sine egne behov. Standarden gir bare muligheten i form av å definere disse som generiske elementer / attributter.

Elementene i ADDML 8.3.

I det etterfølgende er hvert enkelt element i ADDML beskrevet.

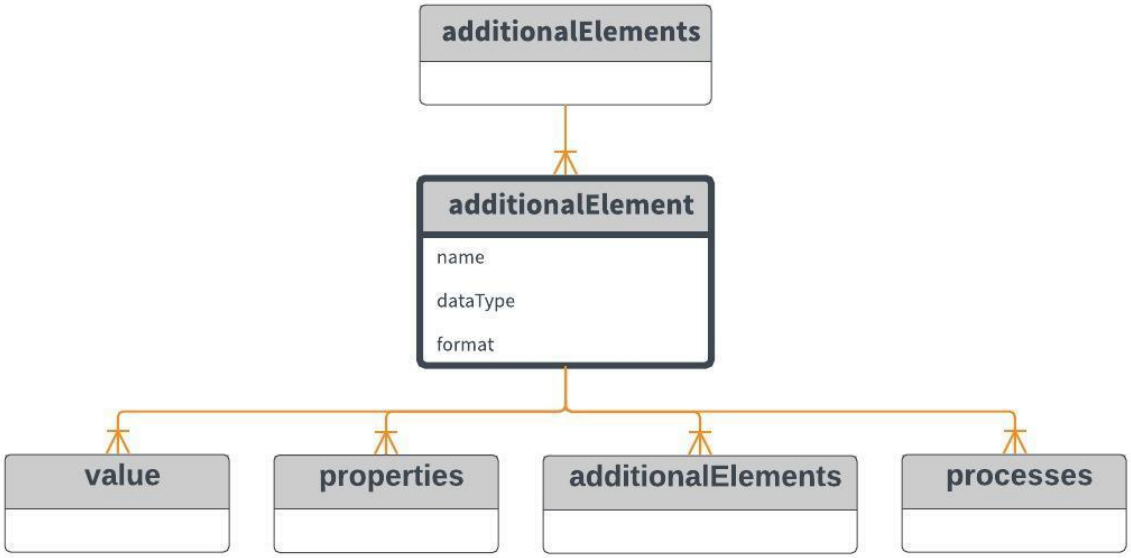
Hvert element har en figur som viser hvilke andre elementer det har en relasjon til, både som overliggende element og eventuelle underliggende element.

I tabellform er det gitt grunnleggende informasjon om elementet. Og deretter i en ny tabell alle attributter som elementet har. Også underliggende element er vist i en tabellform.

Til slutt er det vist et enkelt eksempel med bruk av elementet. For noen element er det vist til eksempler under andre element.

additionalElement

Elementet *additionalElement* utgjør et egetdefinert element. Standarden selv definerer ingen tilleggselementer, men lar det være opp til brukerne å definere sine egne. Sammen med *additionalElements* danner *additionalElement* muligheten for å kunne bygge sin egen generiske struktur.



Elementnavn	Kravtype	Beskrivelse
additionalElement	mandatory	Identifiserer et enkelt tilleggselement. Et tilleggselement defineres av brukeren.

Attributter	Kravtype	Beskrivelse
name	mandatory	Alle tilleggselement må ha et navn. Navnet identifiserer tilleggselementet og vil være utgangspunktet for eventuelle operasjoner som skal utføres på elementet eller gjenfinning av elementet for annet bruk.
dataType	optional	Siden tilleggselementet kan ha et underelement med en verdi, kan man identifisere datatypen til denne verdien.
format	optional	Siden tilleggselementet kan ha et underelement med en verdi, kan man identifisere formatet til denne verdien. Hvilke formater som skal aksepteres må brukeren selv definere.

Underliggende elementer	Kravtype	Forekomster
value	optional	0 - 1
properties	optional	0 - 1
additionalElements	optional	0 - 1

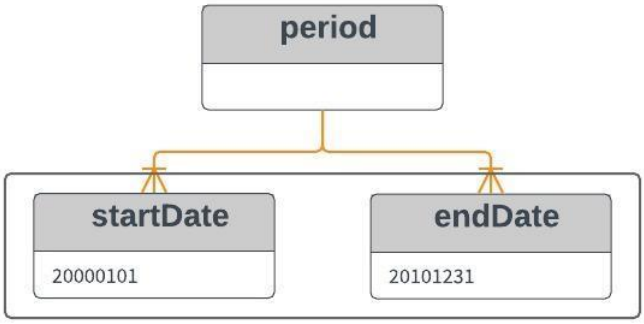
processes	optional	0 - 1

Eksempel:

```
<additionalElement name=”period”>
   <additionalElements>
      <additionalElement name=”startDate” datatype=”date” format=”YYYYMMDD”>
         <value>20000101</value>
      </additionalElement>
      <additionalElement name=”endDate” datatype=”date” format=”YYYYMMDD”>
         <value>20101231</value>
      </additionalElement>
   </additionalElements>
</additionalElement>
```

I eksempelet er det definert et tilleggselement period, som består av to underliggende tilleggselementer – startDate og endDate. Disse to elementene har hver sin verdi som er en dato på formen YYYYMMDD, altså 4-sifret år, 2-sifret måned og 2-sifret dag.

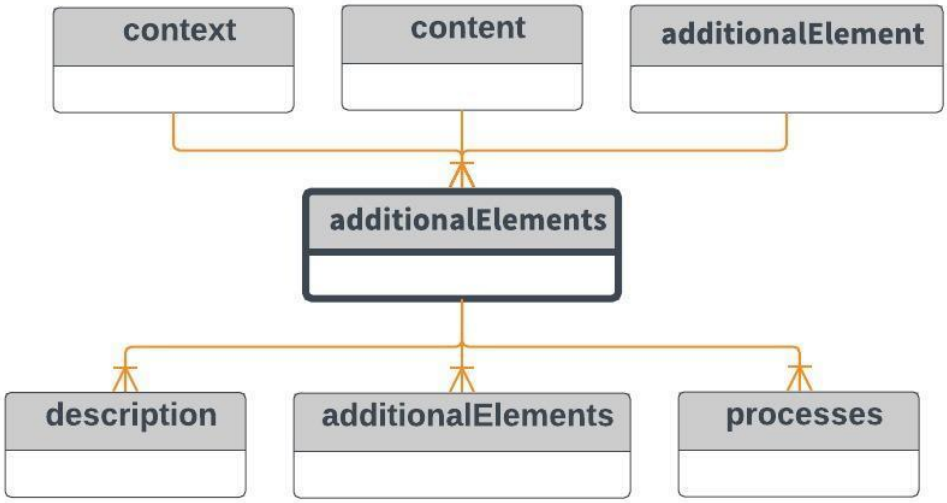
Dette kan også illustreres slik:



Hvor den tomme rammen markerer et omslag rundt en gruppe av elementer og altså tilsvarer nivået additionalElements.

additionalElements

Elementet additionalElements er et samlenivå for gruppering av tilleggselementer.



Elementnavn	Kravtype	Beskrivelse
additionalElements	optional	Dette elementet benyttes for å samle tilleggselementer i grupper. Og derigjennom å kunne danne strukturer av tilleggselementer.

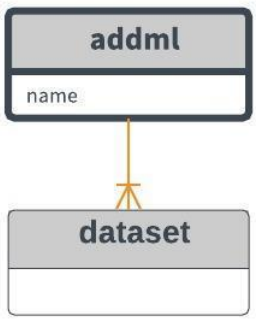
Ingen attributter.

Underliggende elementer	Kravtype	Forekomster
description	optional	0 - 1
additionalElement	optional	0 - n
processes	optional	0 - 1

For eksempel se *additionalElement*.

addml

På toppnivået er det kun ett element – *addml*.

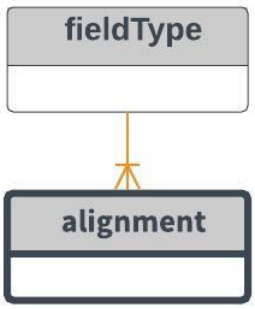


Elementnavn	Kravtype	Beskrivelse
addml	mandatory	*addml* er toppnivået i strukturen. Dette elementet skal eksistere en og kun en gang i henhold til reglene for XML.

Attributter	Kravtype	Beskrivelse
name	mandatory	*name* angir navnet til denne addml-filen. Det er ingen krav til navnsetting i utgangspunktet.

Underliggende elementer	Kravtype	Forekomster
dataset	mandatory	1 - n

alignment



Elementnavn	Kravtype	Beskrivelse
alignment	optional	*alignment* benyttes for å angi hvorvidt innholdet i et felt er venstrejustert, høyrejustert eller midtstilt. Brukeren selv må definere de faktiske verdiene for *alignment*.

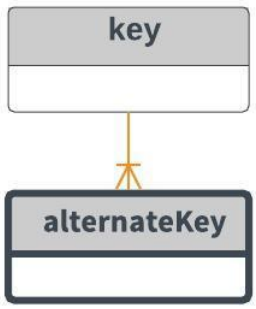
Ingen attributter og ingen underliggende elementer.

Eksempel:

<fieldType name=”streng”>
....
<alignment>***venstre***</alignment>
....
</fieldType>

I eksemplet er felttypen streng definert å være venstrejustert. Selve verdien i elementet kunne f.eks. også vært venstrejustert eller bare v.

alternateKey



Elementnavn	Kravtype	Beskrivelse
alternateKey	optional	Dette elementet benyttes for å angi alternative nøkler. I praksis vil dette gi muligheten for å angi indekser. Selve elementet er kun et flagg.

Ingen attributter og ingen underliggende elementer.

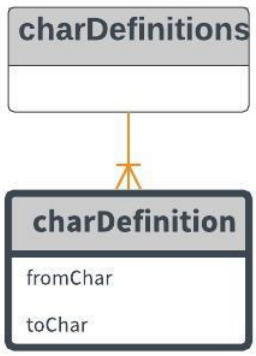
Eksempel:

```
<key name="indeks1">
   <alternateKey/>
   <fieldDefinitionReferences>
      ....
   </fieldDefinitionReferences>
</key>
```

I eksemplet er det definert en nøkkel indeks1 som en alternative nøkkel (evt en indeks) og hvor de elementene som inngår er definert under *fieldDefinitionReferences* (her bare angitt med ….).

charDefinition

charDefinition er ment å kunne benyttes ved spesialtegn som ikke følger vanlig standard. I de fleste tilfeller vil det være nok å bare angi tegnsett. Men dersom et datasett har avvikende tegn kan man benytte denne muligheten til å redefinere de få tegnene som avviker.



Elementnavn	Kravtype	Beskrivelse
charDefinition	mandatory	Elementet benyttes for angivelse av endring av en verdi for ett enkelt tegn som avviker fra tegnsettet som er oppgitt.

Attributter	Kravtype	Beskrivelse
fromChar	mandatory	Angivelse av verdien slik den er definert i tegnsettet. Hva slags type verdi som skal benyttes er det opp til brukeren å bestemme.
toChar	mandatory	Angivelse av verdien slik den er i datasettet og som altså avviker fra tegnsettet. Hva slags type verdi som skal benyttes er det opp til brukeren å bestemme.

Ingen underliggende elementer.

Eksempel:

```
<flatFileType name="fil1">
   <charset>EBCDIC</charset>
   <charDefinitions>
      <charDefinition fromChar="E6" toChar="5B">
      <charDefiniton fromChar="F8" toChar="7C">
      <charDefinition fromChar="E5" toChar="5D">
   </charDefinitions>
   ….
</flatFileType>
```

I eksempelet er vist en mulig endring av verdien av de norske bokstavene æ, ø og å (små verdier) fra deres standard versjon i EBCDIC til en spesiell variant som har vært i bruk i Norge. Nemlig liten æ inn på plassen for [, liten ø inn på plassen for | og liten å inn på plassen for].

I eksemplet er verdiene av tegnene oppgitt med hexadesimal verdi.

charDefinitions

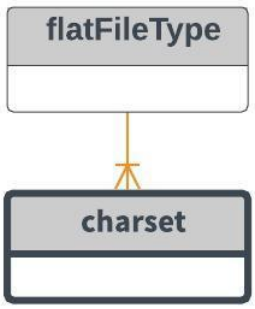
Elementnavn	Kravtype	Beskrivelse
charDefinitions	optional	Dette elementet er et gruppenivå som inneholder de enkelte tegnene som har avvikende verdi.

Ingen attributter.

Underliggende elementer	Kravtype	Forekomster
charDefinition	optional	1 – n

For eksempel se *charDefinition*.

charset



Elementnavn	Kravtype	Beskrivelse
charset	mandatory	Dette elementet benyttes til å angi tegnsettet som benyttes i datasettet. Brukeren må selv definere hvordan de forskjellige tegnsettene skal angis.

Ingen attributter og ingen underliggende elementer.

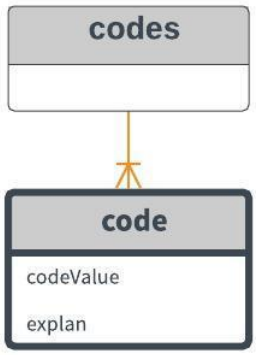
Eksempel:

<flatFileType name=”**fil1**”>
  <charset>***ISO-8859-1***</charset>
  ….
</flatFileType>

I eksemplet er f.eks tegnsettet angitt til å være ISO-8859-1.

code

For et enkelt felt i et datasett kan man definere gyldige kodeverdier. Dette anbefales kun å gjøre dersom det er et begrenset antall, selv om standarden selv ikke setter noen begrensninger på antallet. Men dersom det er mange kodeverdier, anbefales heller at disse følger med datasettet som en egen fil.



Elementnavn	Kravtype	Beskrivelse
code	mandatory	Elementet *code* benyttes til å angi kodeverdier for et enkelt felt i et datasett.

Attributter	Kravtype	Beskrivelse
codeValue	mandatory	Den faktiske gyldige kodeverdien
explan	optional	En forklaring på hva kodeverdien skal bety. Kan være spesielt nyttig når verdien av koden er kort og kryptisk.

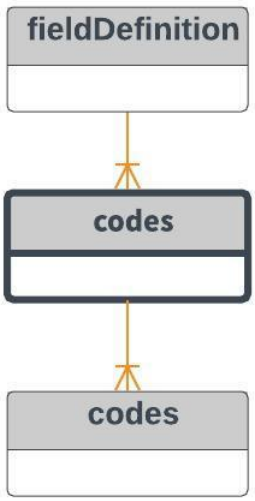
Ingen underliggende elementer.

Eksempel:

```
<fieldDefinition name=”kjønn”>
   ….
   <codes>
      <code codeValue=”M” explan=”Mann”>
      <code codeValue=”K” explan=”Kvinne”>
      <code codeValue=”U” explan=”Ukjent”>
      <code codeValue=” ” explan=”Ikke oppgitt”>
   </codes>
</fieldDefinition>
```

Legg merke til at i eksemplet er blank verdi en gyldig kodeverdi som ikke oppgitt, og at denne er en annen verdi enn ukjent.

codes



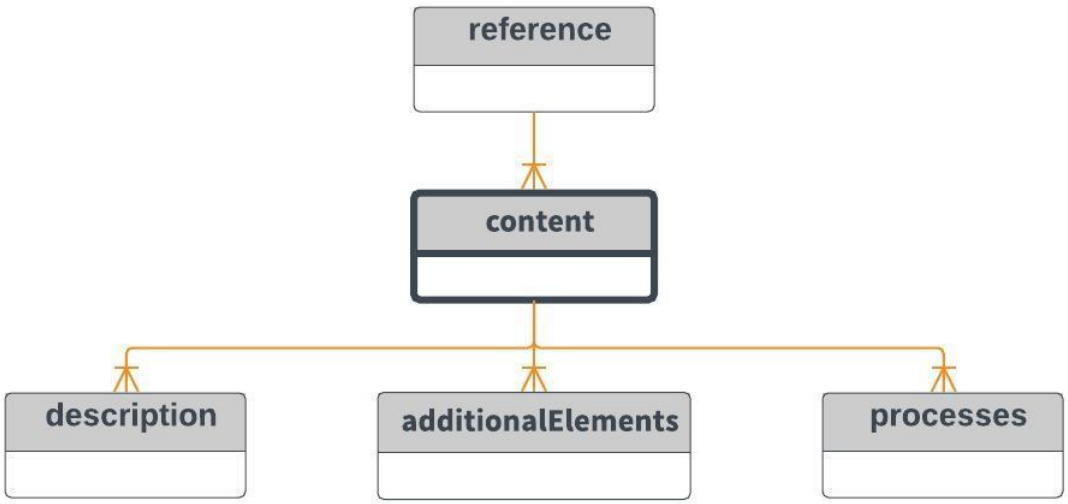
Elementnavn	Kravtype	Beskrivelse
codes	optional	Dette elementet er en gruppering av de gyldige kodene for et felt i datasettet.

Ingen attributter.

Underliggende elementer	Kravtype	Forekomster
code	Optional	1 - n

For eksempel se *code*.

content



Elementnavn	Kravtype	Beskrivelse
content	Optional	*content* inneholder informasjon av innholdsmessig art om avleveringen. Hva slags informasjon som skal være med under *content* må brukeren selv definere ved hjelp av *additionalElement*.

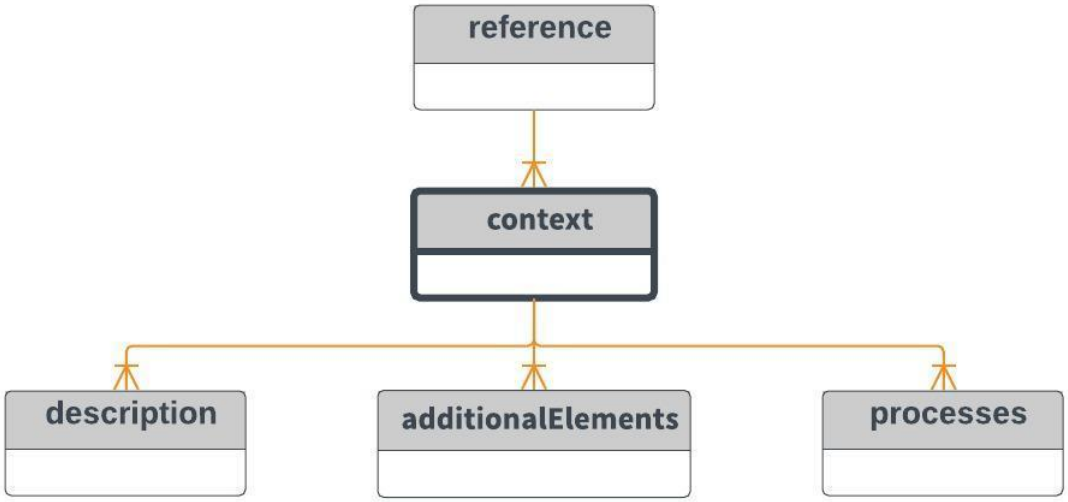
Ingen attributter.

Underliggende elementer	Kravtype	Forekomster
description	optional	0 - 1
additionalElements	optional	0 - 1
processes	optional	0 - 1

Eksempel:

```
<reference>
   <context>
      ….
   </context>
   <content>
      <additionalElements>
         <additionalElement name=”period” >
            <additionalElements>
               ….
            </additionalElements>
         </additionalElement>
      </additionalElements>
   </content>
<reference>
```

context



Elementnavn	Kravtype	Beskrivelse
context	optional	*context* inneholder informasjon av kontekstuell art om avleveringen. Hva slags informasjon som skal være med under *context* må brukeren selv definere ved hjelp av *additionalElement*.

Ingen attributter.

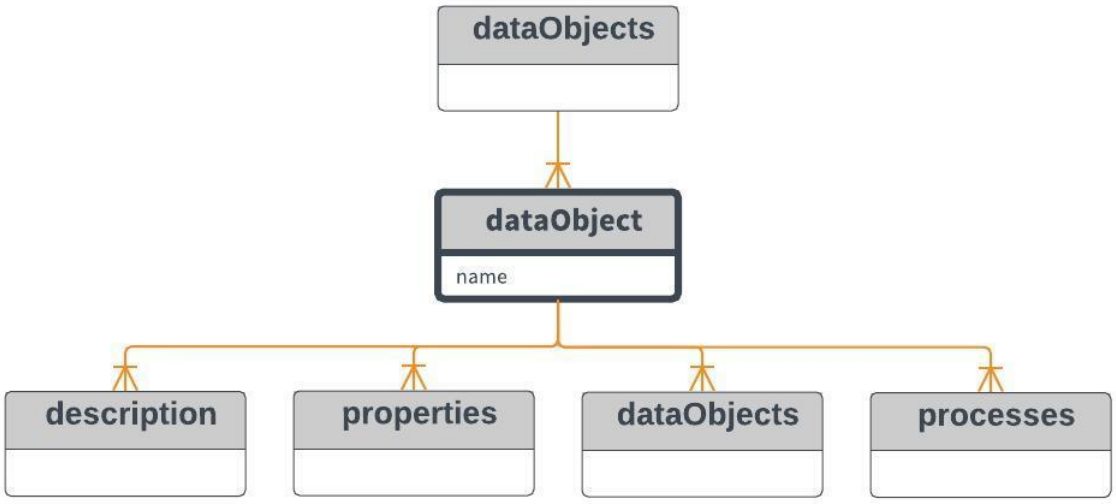
Underliggende elementer	Kravtype	Forekomster
description	optional	0 - 1
additionalElements	optional	0 - 1
processes	optional	0 - 1

Eksempel:

```
<reference>
   <context>
      <additionalElements>
         <additionalElement name=”arkivskaper” datatype=”string”>
            <value>Kulturdepartementet</value>
         </additionalElement>
      </additionalElements>
   </context>
   <content>
      ….
   </content>
<reference>
```

dataObject

De første versjonene av ADDML ble konstruert for å håndtere flate filer. Etter hvert er det også blitt behov for å håndtere andre typer filer, ikke minst xml-filer. Av den grunn ble det innført en ny hoveddel med dataobjekter. Denne delen er generisk, hvor brukeren selv må definere strukturer og informasjonselementer som skal være med.



Elementnavn	Kravtype	Beskrivelse
dataObject	mandatory	Dette er elementer hvor man kan lage en egen hierarkisk struktur for filer som ikke er flate filer. Man må selv bygge opp strukturen på det viset en selv føler blir best mulig.

Attributter	Kravtype	Beskrivelse
name	mandatory	Navnet på dataobjektet som defineres.

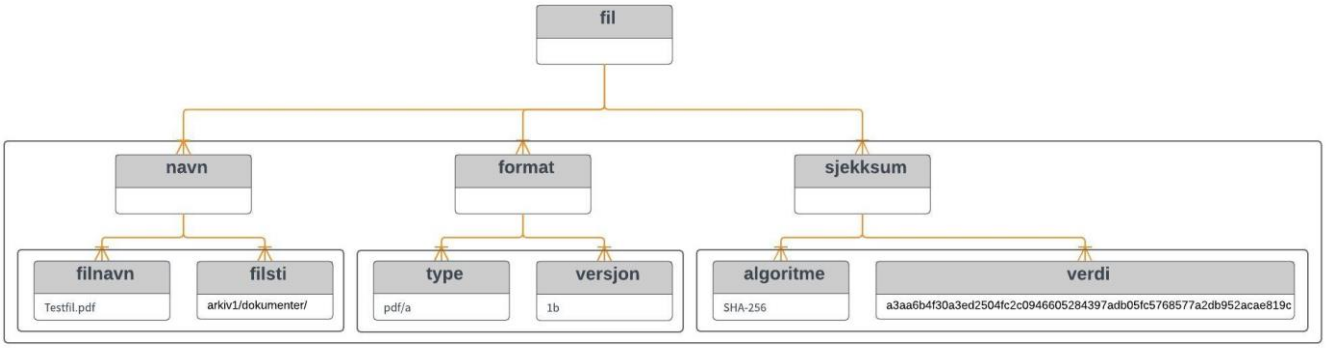
Underliggende elementer	Kravtype	Forekomster
description	optional	0 - 1
properties	optional	0 - 1
dataObjects	optional	0 - 1
processes	optional	0 - 1

Eksempel:

```
<dataObject name=”fil”>
  <dataObjects>
    <dataObject name=”navn”>
      <dataObjects>
        <dataObject name=”filnavn”>Testfil.pdf</dataObject>
        <dataObject name=”filsti”>arkiv1/dokumenter/</dataObject>
      </dataObjects>
    </dataObject>
    <dataObject name=”format”>
      <dataObjects>
        <dataObject name=”type”>pdf/a</dataObject>
        <dataObject name=”versjon”>1b</dataObject>
      </dataObjects>
    </dataObject>
    <dataObject name=”sjekksum”>
      <dataObjects>
        <dataObject name=”algoritme”>SHA-256</dataObject>
        <dataObject name=”verdi”>
            a3aa6b4f30a3ed2504fc2c0946605284397adb05fc5768577a2db952acae819c
        </dataObject>
      </dataObjects>
    </dataObject>
  </dataObjects>
</dataObject>
```

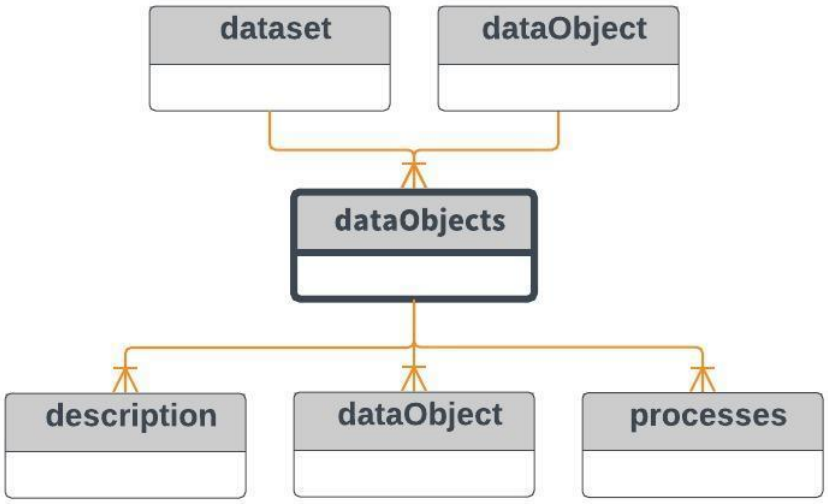
I eksempelet er det definert et dataobjekt som er en fil. Under denne filen er det så definert tre informasjonsobjekter om filen – navn, format og sjekksum. Disse tre har alle tre et nytt undernivå med to informasjonsobjekter for hver. For navn er det definert filnavn – som inneholder selve navnet på filen, og filsti – som inneholder stien. For format er det definert type – som angir hva slags type fil dette er, og versjon – som angir versjonen av typen. For sjekksum er det definert algoritme – som angir hvilken algoritme som er benyttet for å beregne sjekksummen, og verdi – som angir selve sjekksum verdien.

Dette kan også illustreres på følgende måte hvor boksene angir et dataObject og de tomme rammene nivået dataObjects:



dataObjects

De første versjonene av ADDML ble konstruert for å håndtere flate filer. Etter hvert er det også blitt behov for å håndtere andre typer filer, ikke minst xml-filer. Av den grunn ble det innført en ny hoveddel med dataobjekter. Denne delen er generisk, hvor brukeren selv må definere strukturer og informasjonselementer som skal være med.



Elementnavn	Kravtype	Beskrivelse
dataObjects	optional	Dette er elementer hvor man kan lage en egen hierarkisk struktur for filer som ikke er flate filer. Man må selv bygge opp strukturen på det viset en selv føler blir best mulig.

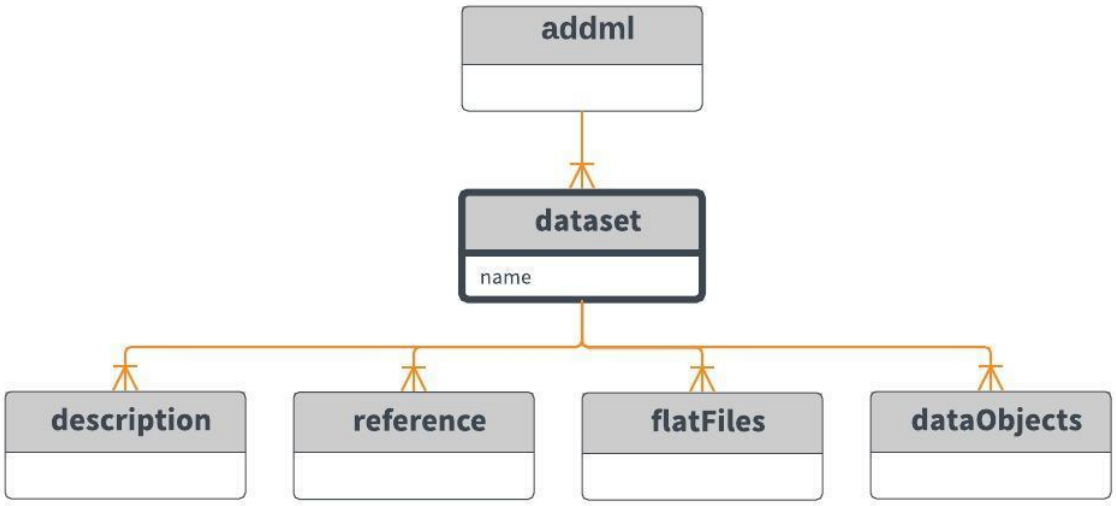
Ingen attributter

Underliggende elementer	Kravtype	Forekomster
description	optional	0 - 1
dataObject	mandatory	1 - n
processes	optional	0 - 1

For eksempel se dataObject.

dataset

Hovednivået dataset som tilsvarer et datasett har også bare ett enkelt element. Til gjengjeld kan dette forekomme flere ganger.



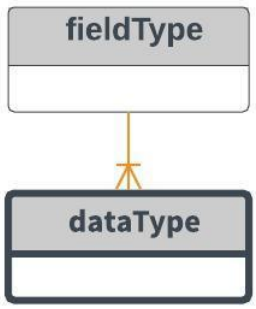
Elementnavn	Kravtype	Beskrivelse
dataset	mandatory	*dataset* er hovednivået i beskrivelsen. I Arkivverket tilsvarer dette et arkivuttrekk. Imidlertid kan en og samme beskrivelse også inneholde flere *dataset*. Dette for at det skal være mulig å samle beskrivelser når de skal benyttes sammen, for eksempel i en brukssituasjon.

Attributter	Kravtype	Beskrivelse
name	optional	*name* angir navnet til det spesifikke datasettet. Det er ingen krav til navnsetting i utgangspunktet, men navnene for dataset må være unike innen en addml-fil.

Underliggende elementer	Kravtype	Forekomster
description	optional	0 - 1
reference	optional	0 - 1
flatFiles	optional	0 - 1
dataObjects	optional	0 - 1

Ved flere datasett i samme addml-fil er det anbefalt å bruke attributten name for hvert datasett for å holde de adskilt.

dataType



Elementnavn	Kravtype	Beskrivelse
dataType	mandatory	Hvilken datatype et felt er. Brukeren må selv definere gyldige datatyper og hvordan de skal betegnes.

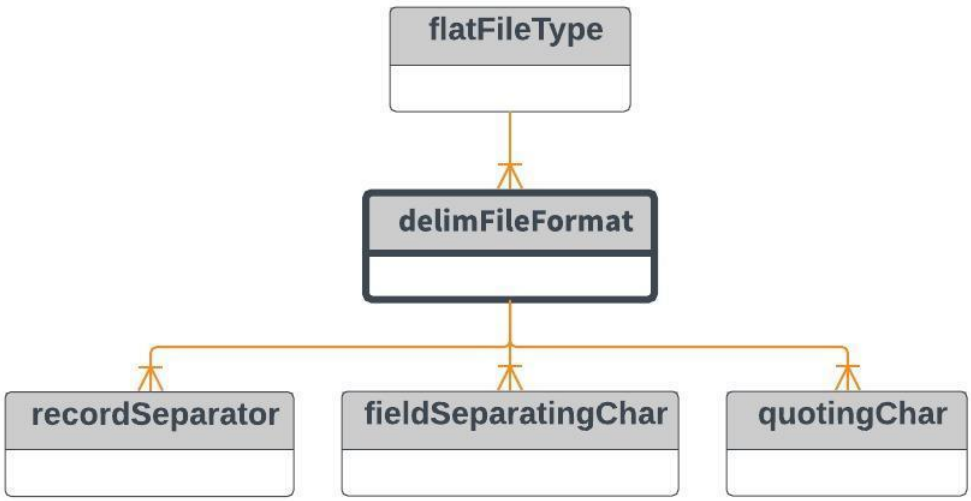
Ingen attributter og ingen underliggende elementer.

Eksempel:

<fieldType name=”**heltall**”>
....
<dataType>***integer***</dataType>
....
</fieldType>

I eksemplet er det definert en felttype som heter heltall. Denne er da definert som datatypen integer (integer for heltall).

delimFileFormat



Elementnavn	Kravtype	Beskrivelse
delimFileFormat	optional	Dette elementet benyttes for flate filer for å angi at feltene i filene er atskilt med et tegn og ikke har faste posisjoner.

Ingen attributter.

Underliggende elementer	Kravtype	Forekomster
recordSeparator	mandatory	1
fieldSeparatingChar	mandatory	1
quotingChar	optional	0 - 1

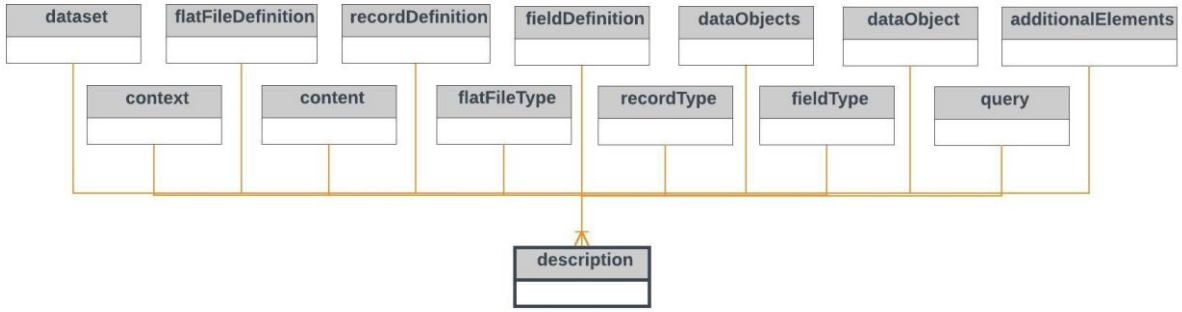
Eksempel:

<flatFileType name=”**fil1**”>
….
<delimFileFormat/>
</flatFileType>

I eksemplet er fil1 oppgitt å være en fil inneholdende felter som er separert ved et skilletegn, og dermed ikke har noen fast posisjon.

description

Elementet beskrivelse benyttes flere steder i strukturen.



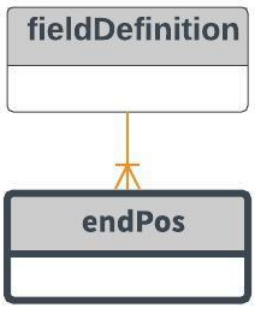
Elementnavn	Kravtype	Beskrivelse
description	optional	Dette elementet inneholder en tekstlig beskrivelse knyttet til elementet det ligger under.

Ingen attributter og ingen underelementer.

Eksempel:

<dataset>
  <description>
    Dette datasettet inneholder et arkivuttrekk fra system x fra arkivskaper y.
  </description>
  ….
</dataset>

endPos



Elementnavn	Kravtype	Beskrivelse
endPos	optional	Dette elementet inneholder sluttposisjonen for et felt når det er snakk om et uttrekk med en flat fil med fast posisjonering.

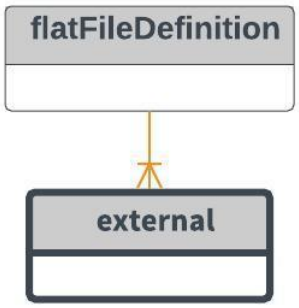
Ingen attributter og ingen underliggende elementer.

For eksempel se fieldDefinition.

external

I noen tilfeller er det behov for å knytte et datauttrekk til et annet. Det er da definert to elementer i ADDML som er tenkt å benyttes til dette formålet. Elementene er *external* som angir at den filen som her defineres ikke er med i selve datauttrekket, og *incomplete* som angir at definisjonen ikke er komplett. Tanken er at man for eksterne filer bare definerer de elementene som er nødvendig for å opprette referanser mellom filen som defineres utenfra og de interne filene.

Et eksempel på en slik kobling kan være at man i uttrekket definerer postkatalogen som en ekstern fil. Dette fordi den benyttes av flere systemer samtidig. Samtidig er det også opprettet referanser (nøkler) fra interne elementer med postnr til denne eksterne postkatalogen.



Elementnavn	Kravtype	Beskrivelse
external	optional	Dette elementet benyttes når det er referanser til filer som ikke er med i uttrekket. Elementet er da et flagg som benyttes for å angi at filen ikke er med.

Ingen attributter og ingen underliggende elementer.

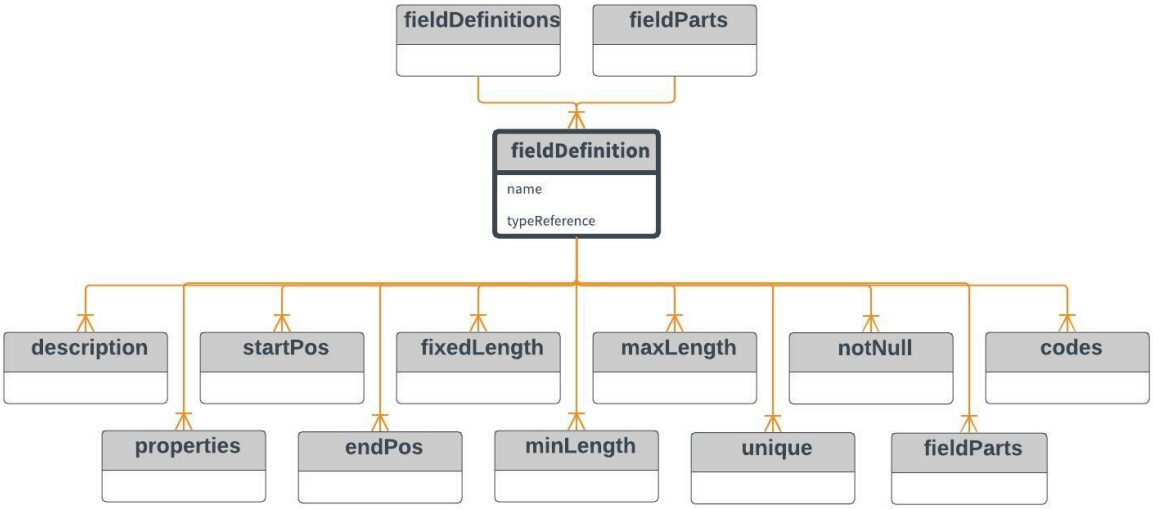
Eksempel:

```
<flatFileDefinition name”fil1”>
   <recordDefinitions>
      <recordDefinition name=”post1”>
         <keys>
            <key>
               <foreignKey>
                  <flatFileDefinitionReference name=”fil2”>
                     <recordDefinitionReferences>
                        <recordDefinitionReference name=”post2”>
                           <fieldDefinitionReferences>
                              <fieldDefinitionReference name=”postnr”/>
                           </fieldDefinitionReferences>
                        </recordDefinitionReference>
                     </recordDefinitionReferences>
                  </flatFileDefinitionReference>
                  <relationType>n:1</relationType>
               </foreignKey>
               <fieldDefinitionReferences>
                  <fieldDefinitionReference name=”postnr”>
               </fieldDefinitionReferences>
            </key>
         </keys>
         <fieldDefinitions>
            <fieldDefinition name=”navn”>
               ….
            </fieldDefinition>
            ….
            <fieldDefinition name=”postnr”>
               ….
            </fieldDefinition>
         </fieldDefinitions>
      </recordDefinition>
   </recordDefinitions>
</flatFileDefinition>
<flatFileDefinition name=”fil2”>
   <external/>
   <recordDefinition name=”post2”>
      <incomplete>
      <fixedLength>244</fixedLength>
      <fieldDefinition name=”postnr”>
         <startPos>1</startPos>
         <endPos>4</endpos>
      </fieldDefinition>
   </recordDefinition>
</flatFileDefinition>
```

I eksemplet er det definert en fil (fil1) med en post (post1) hvor det er definert flere felt, bl.a. navn og postnr. Samt definert en ufullstendig og ekstern fil (fil2) med en post (post2) med et felt – postnr. Samt definert en fremmednøkkel fra fil1 til fil2 med postnr som nøkkelfelt.

fieldDefinition

I ADDML er det tre parallelle informasjonstyper, den øverste er den generelle typen, hvor basis informasjon om felter defineres. Deretter kommer definisjonen av det enkelt felt, hvor man for mer eksplisitt informasjon om et felt og til slutt det fysiske delen som dog bare er på filnivå. Elementet som forklares her er på definisjonslaget.



Elementnavn	Kravtype	Beskrivelse
fieldDefinition	mandatory	Dette elementet er en beholder for informasjon om ett enkelt felt for filer som er i fast format, dvs. fast posisjonering eller tegnseparert.

Attributter	Kravtype	Beskrivelse
name	mandatory	Navn på feltet som defineres.
typeReference	mandatory	Referanse til typen felt som defineres.

Underliggende elementer	Kravtype	Forekomster
description	optional	0 - 1
properties	optional	0 - 1
startPos	optional	0 - 1
endPos	optional	0 - 1
fixedLength	optional	0 - 1
minLength	optional	0 - 1
maxLength	optional	0 - 1
unique	optional	0 - 1
notNull	optional	0 - 1

fieldParts	optional	0 - 1
codes	optional	0 - 1

Eksempel:

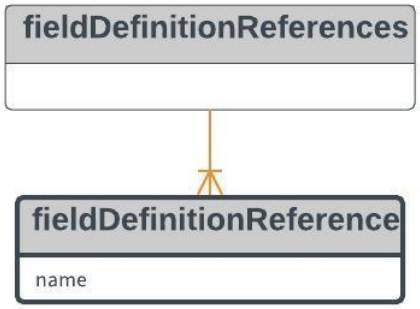
<fieldDefinition name=”**kjønn**” typeReference=”**flagg**”>
<startPos>***1***</startPos>
<endPos>***1***</endPos>
<codes>
<code codeValue=”**M**” explan=”**Mann**”>
<code codeValue=”**K**” explan=”**Kvinne**”>
<code codeValue=”**U**” explan=”**Ukjent**”>
<code codeValue=” ” explan=”**Ikke oppgitt**”>
</codes>
</fieldDefinition>

eller:

<fieldDefinition name=”**fødselsnr**” typeReference=”**fnr**”>
<startPos>***1***</startPos>
<fixedLength>***11***</ fixedLength >
<notNull/>
<unique/>
</fieldDefinition>

Over er vist et par eksempler med felter kjønn og fødselsnr.

fieldDefinitionReference



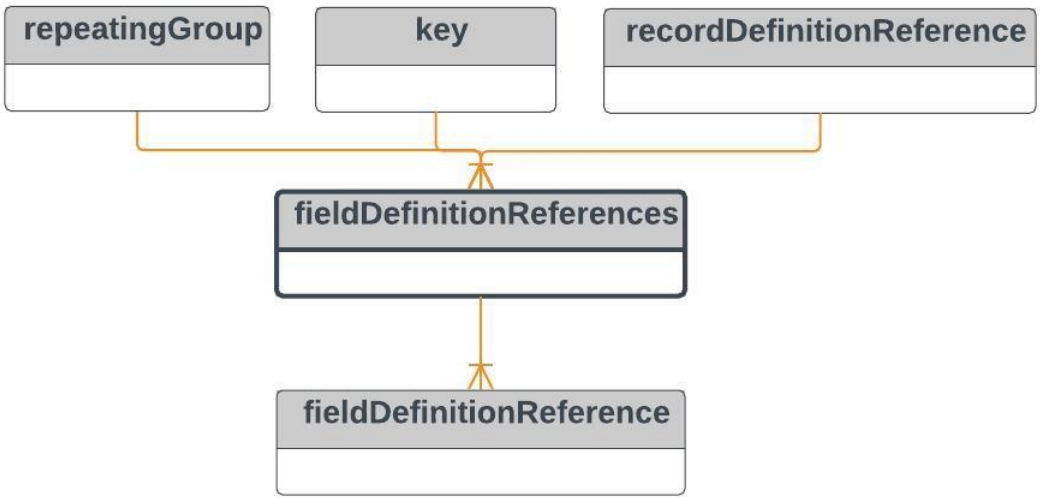
Elementnavn	Kravtype	Beskrivelse
fieldDefinitionReference	mandatory	Referanse til et felt

Attributter	Kravtype	Beskrivelse
name	mandatory	Navn på feltet det refereres til

Ingen underliggende elementer.

For eksempel se *external*.

fieldDefinitionReferences



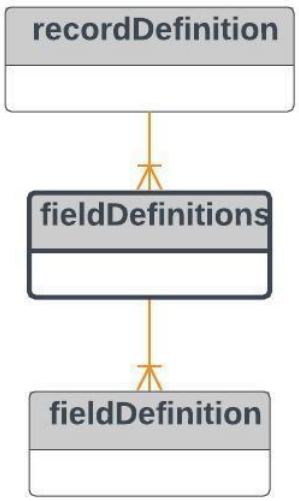
Elementnavn	Kravtype	Beskrivelse
fieldDefinitionReferences	optional	Dette elementet er en samling av feltreferanser. Antallet kan være fra 1 til mange referanser.

Ingen attributter.

Underliggende elementer	Kravtype	Forekomster
fieldDefinitionReference	mandatory	1 - n

For eksempel se *external*.

fieldDefinitions



Elementnavn	Kravtype	Beskrivelse
fieldDefinitions	mandatory	Elementet er en overordnet samling for de enkelt feltdefinisjonene for en post.

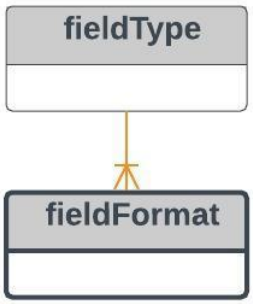
Ingen attributter.

Underliggende elementer	Kravtype	Forekomster
fieldDefinition	mandatory	1 - n

Eksempel:

<recordDefinition name=”**post1**”>
….
<fieldDefinitions>
<fieldDefinition name=”**felt**”>
….
</fieldDefinition>
<fieldDefinition name=”**felt2**”>
….
</fieldDefinition>
….
</fieldDefinitions>
</recordDefinition>

fieldFormat



Elementnavn	Kravtype	Beskrivelse
fieldFormat	optional	Elementet benyttes for å angi formatet for et felt. For visse datatyper vil det være unaturlig å benytte elementet, mens for andre vil det nesten være et krav. Hvilket betyr at dette elementet må sees i sammenheng med elementet *dataType*.

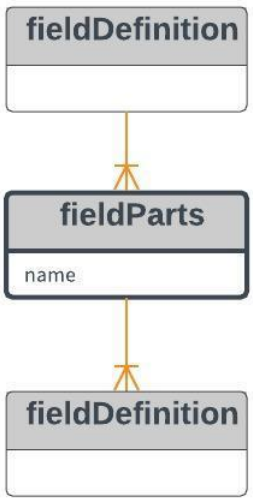
Ingen attributter og ingen underliggende elementer.

Eksempel:

<fieldType name="**dato8**">
  <dataType>***date***</dataType>
  <fieldFormat>***ÅÅÅÅMMDD***</fieldFormat>
  ….
</fieldType>

Eksemplet viser hvordan man kan definere et datofelt med formatet ÅÅÅÅMMDD, dvs som 4-sifret år, 2-sifret måned og 2-sifret dag.

fieldParts



Elementnavn	Kravtype	Beskrivelse
fieldParts	optional	I noen tilfeller er det ønskelig å kunne dele opp et element i mindre deler, samtidig som man også vil ha muligheten til å referere til det hele. Elementet *fieldParts* er ment å dekke dette behovet.

Attributter	Kravtype	Beskrivelse
name	optional	

Underliggende elementer	Kravtype	Forekomster
fieldDefinition	mandatory	1 - n

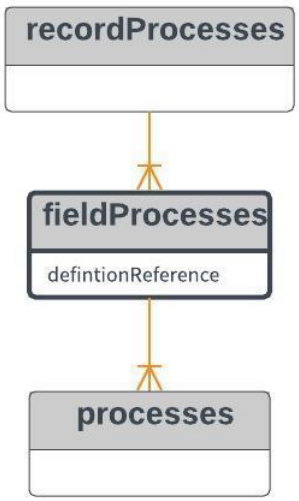
Eksempel:

```
<fieldDefinition name=”dokumentdato” typeReference=”dato8”>
   ….
   <fieldParts name=”dokumentdato”>
      <fieldDefinition name=”dokumentaar” typeReference=”heltall4”>
         ….
      </fieldDefinition>
      <fieldDefinition name=”dokumentmaaned” typeReference=”heltall2”>
         ….
      </fieldDefinition>
      <fieldDefinition name=”dokumentdag” typeReference=”heltall2”>
         ….
      </fieldDefinition>
   </fieldParts>
</fieldDefinition>
```

Eksemplet viser hvordan et datofelt kan defineres med delfeltene år, måned og dag. Her kunne man også tenke seg å bygge eksemplet ut slik at man hadde en del for dato og en del for klokkeslett av et felt, og at disse igjen hadde delfeltene år, måned, dag for datodelen og time, minutt, sekund for klokkedelen.

Vær også oppmerksom på at selv om et element har *fieldParts*, trenger man ikke definere alle delene. I eksemplet over kunne man kanskje bare være interessert i året og kun det.

fieldProcesses



Elementnavn	Kravtype	Beskrivelse
fieldProcesses	optional	Disse elementene representerer prosesser på de forskjellige delene av strukturen i en ADDML-fil som beskriver flate filer. Prosess-strukturen følger samme struktur som strukturen med flat fil-, post- og feltdefinisjonene. Selve prosessangivelsene knyttes til ønsket sted i prosess-strukturen

Attributter	Kravtype	Beskrivelse
definitionReference	mandatory	

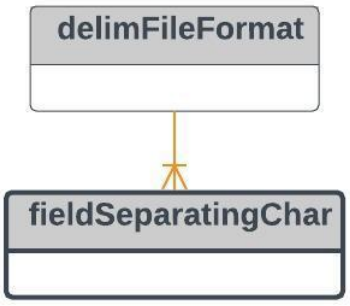
Underliggende elementer	Kravtype	Forekomster
processes	mandatory	1

Eksempel:

```
<recordProcesses definitionReference=”post1”>
   …
   <fieldProcesses definitionReference=”personnr”>
      <processes>
         <process name=”Control_Birthno”/>
         …
      </processes>
   </fieldProcesses>
   …
</recordProsesses>
```

Eksemplet viser hvordan man kan angi kontroll av et fødselsnr i et element (felt) med navn persnr. Den midterste av … angir at det her kan være flere prosesser knyttet til elementet persnr.

fieldSeparatingChar



Elementnavn	Kravtype	Beskrivelse
fieldSeparatingChar	mandatory	Elementet benyttes til å angi hva slags tegn eller kombinasjon av tegn som benyttes som skilletegn mellom felter (elementer) i en post.

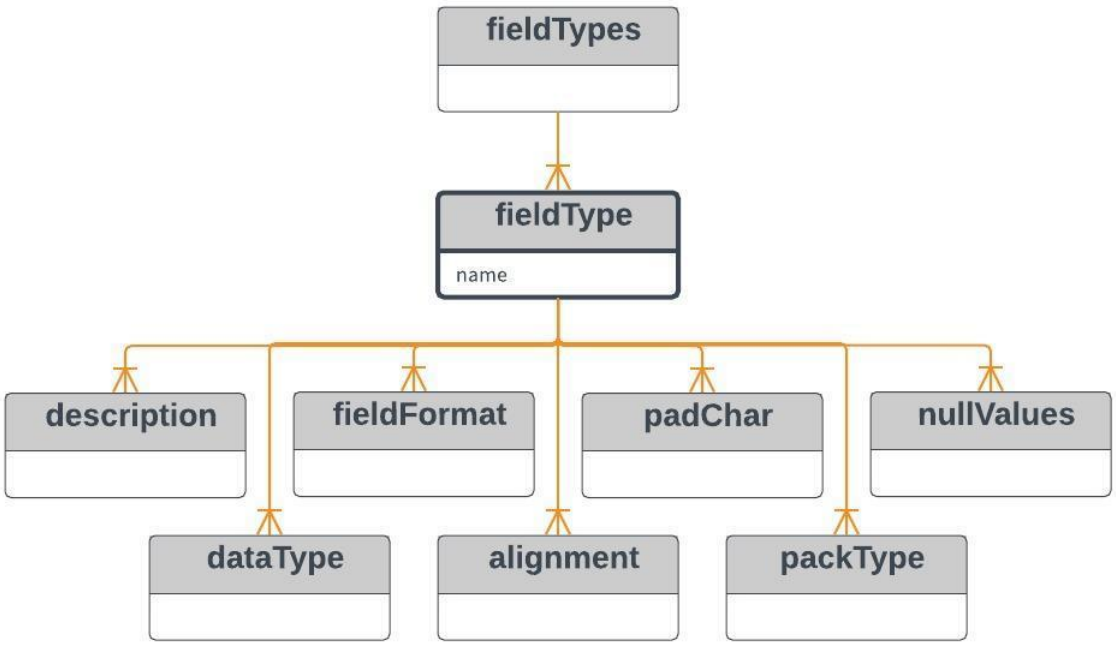
Ingen attributter og ingen underliggende elementer.

Eksempel:

<flatFileType name=”**fil1**”>
….
<delimFileFormat>
<fieldSeparatingChar>***;***<fieldSeparatingChar>
</delimFileFormat>
</flatFileType>

I eksemplet er angitt en fil med tegnseparering og hvor feltskilletegn er semikolon (;).

fieldType



Elementnavn	Kravtype	Beskrivelse
fieldType	mandatory	For typene på feltnivå benyttes elementene *fieldTypes* og *fieldType*, og som vanlig med et samlenivå og et detaljnivå. På feltnivå kan det angis et navn på typen, en beskrivelse (hvis man ønsker), samt datatypen for feltet, feltformatet, hvilken justering feltet har, om feltet er fylt ut med blanke tegn, hva slags pakking feltet har (dersom det er pakket) og om det er benyttet spesielle tegn for nullverdier.

Attributter	Kravtype	Beskrivelse
name	mandatory	

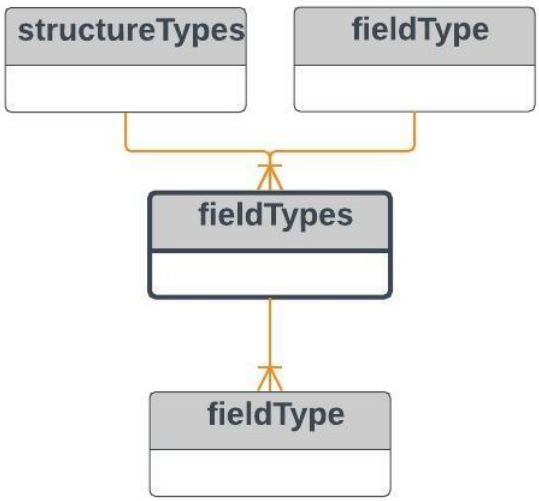
Underliggende elementer	Kravtype	Forekomster
description	optional	0 - 1
dataType	mandatory	1
fieldFormat	optional	0 - 1
alignment	optional	0 - 1
padChar	optional	0 - 1
packType	optional	0 - 1
nullValues	optional	0 - 1

Eksempel:

```
<fieldTypes>
   ….
   <fieldType name=”dato8”>
      <dataType>date</dataType>
      <fieldFormat>YYYY.MM.DD</fieldFormat>
   </fieldType>
   ….
</fieldTypes>
```

I eksemplet er definert en felttype ved navn dato8. Felttypen angir at et felt av denne typen er av dataformat dato (date) og at datoen skal være på formen YYYY.MM.DD som f.eks. 2018.06.28.

fieldTypes



Elementnavn	Kravtype	Beskrivelse
fieldTypes	mandatory	Samlelementet for *fieldType*.

Ingen attributter.

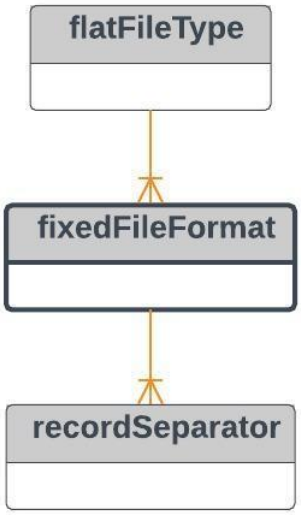
Underliggende elementer	Kravtype	Forekomster
fieldType	mandatory	1 - n

Eksempel:

```
<structureTypes>
   <flatFileTypes>
      ….
   </flatFileTypes>
   <recordTypes>
      ….
   </recordTypes>
   <fieldTypes>
      <fieldType name=”tekst1”>
         ….
      </fieldType>
      <fieldType name=”dato8”>
         ….
      </fieldType>
      <fieldType name=”heltall1”>
         ….
      </fieldType>
      ….
   </fieldTypes>
</structureTypes>
```

I eksemplet er beskrevet tre felttyper (tekst1, dato8 og heltall1) inne i den felles samleangivelsen felttyper (fieldTypes). Det er også angitt at det kan være ytterligere felttyper definert.

fixedFileFormat



Elementnavn	Kravtype	Beskrivelse
fixedFileFormat	optional	Elementet benyttes for å angi at en fil inneholder felter (elementer) med fast posisjonering.

Ingen attributter.

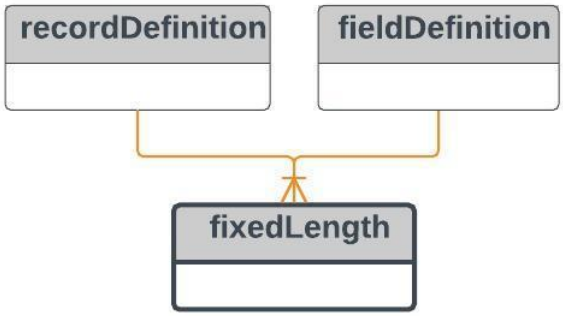
Underliggende elementer	Kravtype	Forekomster
recordSeparator	optional	0 - 1

Eksempel:

<flatFileType name=”**fil1**”>
….
<fixedFileFormat/>
</flatFileType>

I eksemplet er angitt en filtype (fil1) som har fast feltposisjonering.

fixedLength



Elementnavn	Kravtype	Beskrivelse
fixedLength	optional	Elementet benyttes for å angi lengden på en post eller et felt (element) som har en gitt lengde.

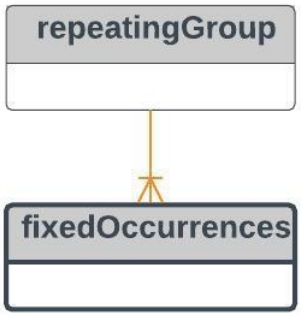
Ingen attributter og ingen underliggende elementer.

Eksempel:

```
<recordDefinition name=”sak” typeReference=”post1”>
   <fixedLength>280</fixedLength>
   <fieldDefinition name=”fnr” typeReference=”streng1”>
      ….
   </fieldDefinition>
</recordDefinition>
```

I eksemplet er angitt en post ved navn sak med fast postlengde på 280 tegn.

fixedOccurrences



Elementnavn	Kravtype	Beskrivelse
fixedOccurrences	optional	Elementet benyttes til å angi antall forekomster av en repeterende gruppe av elementer.

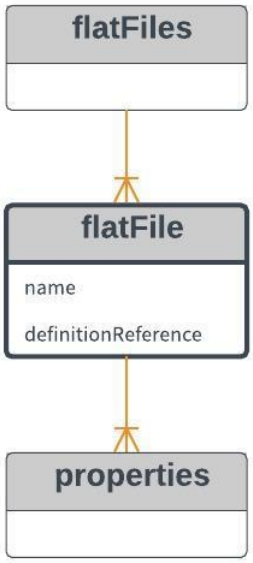
Ingen attributter og ingen underliggende elementer.

Eksempel:

<repeatingGroup name=”**kontingent**”>
  <fixedOccurrences>***5***</fixedOccurrences>
  <fieldDefinitionReferences>
    ….
  </fieldDefinitionReferences>
</repeatingGroup>

I eksemplet er angitt en repeterende gruppe kalt kontingent med en eller flere felt gjentatt som en gruppe 5 ganger.

flatFile



Elementnavn	Kravtype	Beskrivelse
flatFile	mandatory	*flatFiles* er en overbygging av filstrukturen. De enkelte filene gjenfinnes i *flatFile*, mens *flatFiles* samler de sammen til en enhet. *flatFile* inneholder informasjon om en enkelt fil på det fysiske planet. Det finnes her referanse til *flatFileDefinition*, en referanse som kan være mange til en.

Attributter	Kravtype	Beskrivelse
name	mandatory	
definitionReference	mandatory	

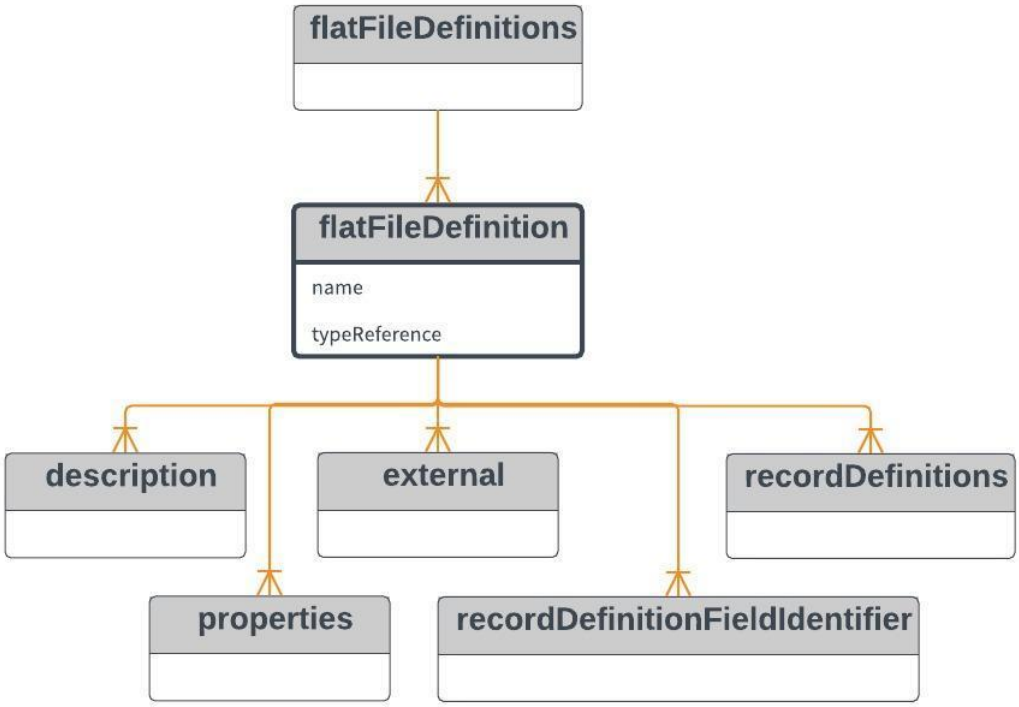
Underliggende elementer	Kravtype	Forekomster
properties	optional	0 - 1

Eksempel:

```
<flatFiles>
   <flatFile name=”sak” definitionReference=”flatfil1”/>
   ….
</flatFiles>
```

I eksemplet er angitt en fil ved navn sak som følger filtypen flatfil1.

flatFileDefinition



Elementnavn	Kravtype	Beskrivelse
flatFileDefinition	mandatory	Elementet samler alle elementer som benyttes for å beskrive en fil. Mer generelle egenskaper for filen er samlet i typebeskrivelsen, mens her er det snakk om elementer som er mer situasjonsspesifikke.

Attributter	Kravtype	Beskrivelse
name	mandatory	
typeReference	optional	

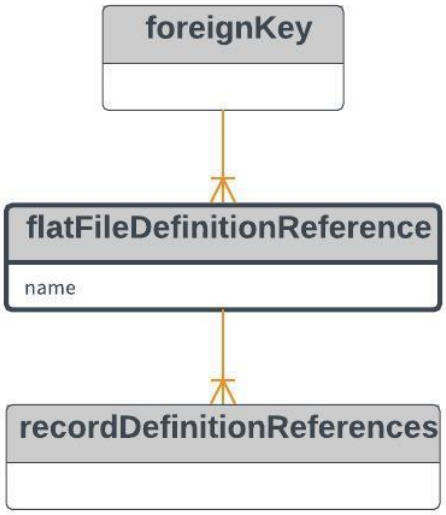
Underliggende elementer	Kravtype	Forekomster
description	optional	0 - 1
properties	optional	0 – 1
external	optional	0 – 1
recordDefinitionFieldIdentifier	optional	0 – 1
recordDefinitions	mandatory	1

Eksempel:

```
<flatFileDefinitions>
   <flatFileDefinition name=”noarksak” typeReference=”fil1”>
      <properties>
         ….
      </properties>
      <recordDefinitions>
         ….
      </recordDefinitions>
   </flatFileDefinition>
</flatFileDefinitions>
```

I eksempelet er det definer en fil ved navn noarksak. Denne har noen egenskaper som ikke er angitt i detalj, men indikert ved properties. Tilsvarende er det også definert poster i filen ved recordDefinitions uten at heller disse er beskrevet i detalj.

flatFileDefinitionReference



Elementnavn	Kravtype	Beskrivelse
flatFileDefinitionReference	mandatory	Elementet benyttes til å angi en referanse til en annen fil i en fremmednøkkel.

Attributter	Kravtype	Beskrivelse
name	mandatory	

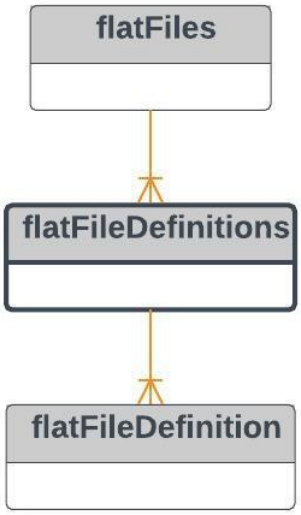
Underliggende elementer	Kravtype	Forekomster
recordDefinitionReferences	optional	0 - 1

Eksempel:

```
<foreignKey>
   <flatFileDefinitionReference name=”noarksak”>
      <recordDefinitionReferences>
         ….
      </recordDefinitionReferences>
   </flatFileDefinitionReference>
   <relationType>1:n</relationType>
</foreignKey>
```

I eksempelet er angitt en fremmednøkkel som henviser til elementer I den definerte filen noarksak. For at dette skal være gyldig må det være definert en fil med navn noarksak i samme addml fil.

flatFileDefinitions



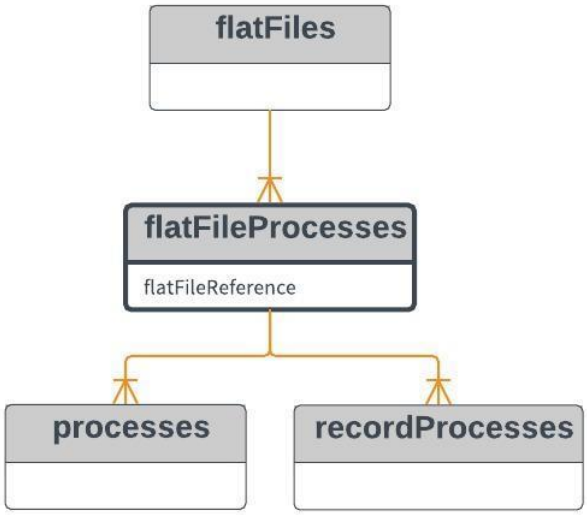
Elementnavn	Kravtype	Beskrivelse
flatFileDefinitions	mandatory	Som med beskrivelsen av den fysiske siden, er det også på det overordnede planet en oppdeling av definisjonsnivået i et samleelement og et definert element for hver fil. Disse elementene er *flatFileDefinitions* og *flatFileDefinition* respektive. Fra *flatFileDefinition* finnes det en referanse til *flatFileType*. Se også *flatFileDefinition*.

Ingen attributter.

Underliggende elementer	Kravtype	Forekomster
flatFileDefinition	mandatory	1 - n

For eksempel se under flatFileDefinition.

flatFileProcesses



Elementnavn	Kravtype	Beskrivelse
flatFileProcesses	optional	Elementet er et samleelement for prosesser som angis, både for prosesser for filnivået og samleelementet for postnivå.

Attributter	Kravtype	Beskrivelse
flatFileReference	mandatory	

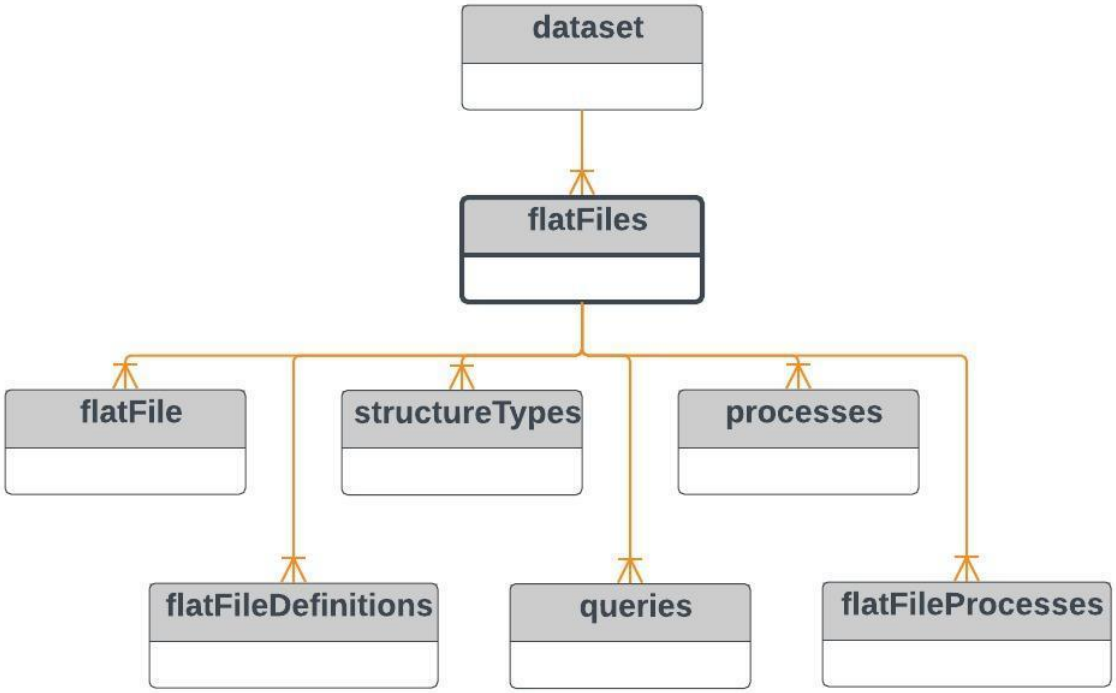
Underliggende elementer	Kravtype	Forekomster
processes	optional	0 - 1
recordProcesses	optional	0 - n

Eksempel:

```
<flatFiles>
   …
   <flatFileProcesses flatFileReference=”noarksak”>
      <processes>
         <process name=”Control_NumberOfRecords”/>
         …
      </processes>
      </recordProsesses>
         …
      </recordProsesses>
   </flatFileProcesses>
   …
</flatFiles>
```

I eksempelet vises hvordan man kan angi en prosess – Control_NumberOfRecords – som skal utføres for filen som heter noarksak. Det forutsettes at en flat fil ved navn noarksak er definert i samme ADDML fil.

flatFiles



Elementnavn	Kravtype	Beskrivelse
flatFiles	mandatory	*flatFiles* er en overbygging av filstrukturen. De enkelte filene gjenfinnes i *flatFile*, mens *flatFiles* samler de sammen til en enhet. *flatFile* inneholder informasjon om en enkelt fil på det fysiske planet. Det finnes her referanse til *flatFileDefinition*, en referanse som kan være mange til en. Se også *flatFile*.

Ingen attributter.

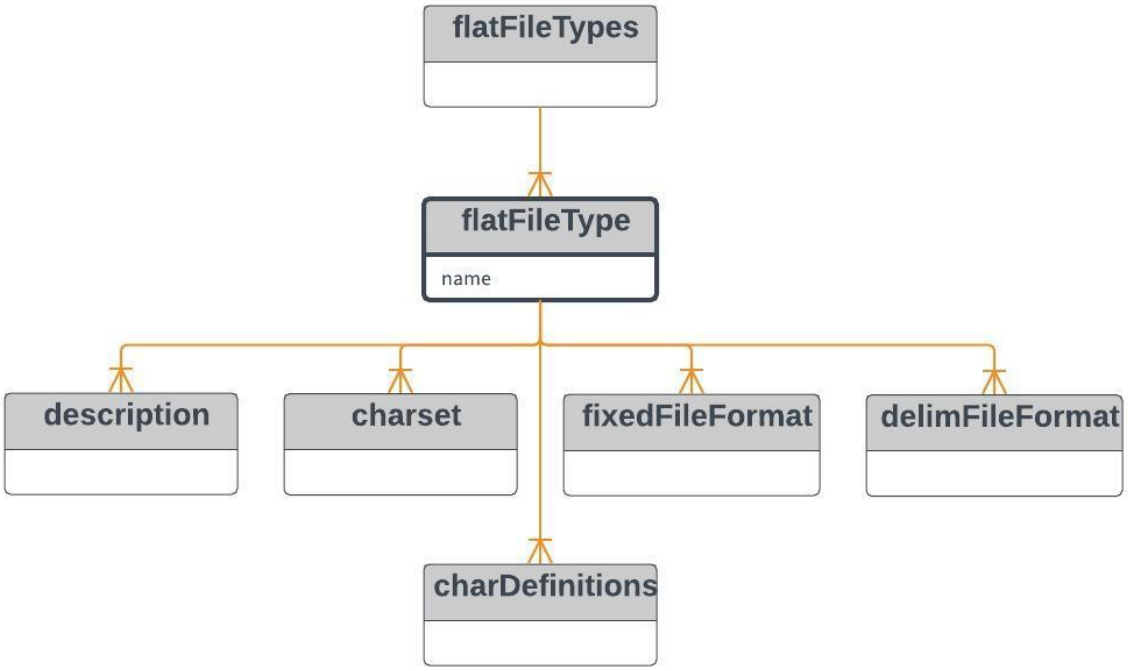
Underliggende elementer	Kravtype	Forekomster
flatFile	optional	0 - n
flatFileDefinitions	mandatory	1
structureTypes	mandatory	1
queries	optional	0 - 1
processes	optional	0 - 1
flatFileProcesses	optional	0 - n

Eksempel:

```
<flatFiles>
   <flatFile name=”sak” definitionReference=”flatfil1”/>
   ….
   <flatFileDefinitions>
      ….
   </flatFileDefinitions>
   ….
   <structureTypes>
      ….
   </structureTypes>
   <processes>
      ….
   </processes>
   <flatFileProcesses>
      ….
   </flatFileProcesses>
</flatFiles>
```

Eksempelet viser elementet med en underliggende fil sak og indikasjon på at det er flere filer. Dessuten starten på en del av de andre elementene som hører til innunder flatFiles.

flatFileType



Elementnavn	Kravtype	Beskrivelse
flatFileType	mandatory	For typene på filnivå benyttes elementene *flatFileTypes* og *flatFileType* – ett samlenivå og ett detaljnivå som vanlig. På filnivå kan det angis et navn på typen, en beskrivelse (hvis man ønsker), samt hvilket karaktersett som er benyttet og om filen er i fast format eller tegnseparert format. Typenivået inneholder elementer som er av mer generell karakter for et sett av filer, i motsetning til definisjonsnivået som inneholder elementer med mer spesifikk informasjon om den enkelte fil.

Attributter	Kravtype	Beskrivelse
name	mandatory	

Underliggende elementer	Kravtype	Forekomster
description	optional	0 - 1
charset	mandatory	1
charDefinitions	optional	0 - 1
fixedFileFormat	optional	0 - 1
delimFileFormat	optional	0 - 1

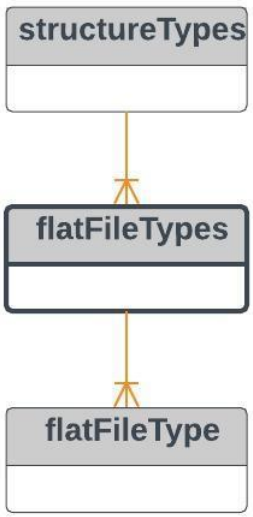
NB!	Når det gjelder de underliggende elementene «fixedFileFormat» og «delimFileFormat» er de i utgangspunktet optional. Dog må en og kun en av de være tilstede.

Eksempel:

```
<flatFileType name=”flatfil1”>
   <charset type=”ISO-8859-1”/>
   <fixedFileFormat/>
</flatFileType>
```

Eksempelet viser en enkel definisjon av en flatFileType med tegnsett ISO-8859-1 og fast format uten skilletegn.

flatFileTypes



Elementnavn	Kravtype	Beskrivelse
flatFileTypes	mandatory	Dette er samleelementet for typer av filer.

Ingen attributter.

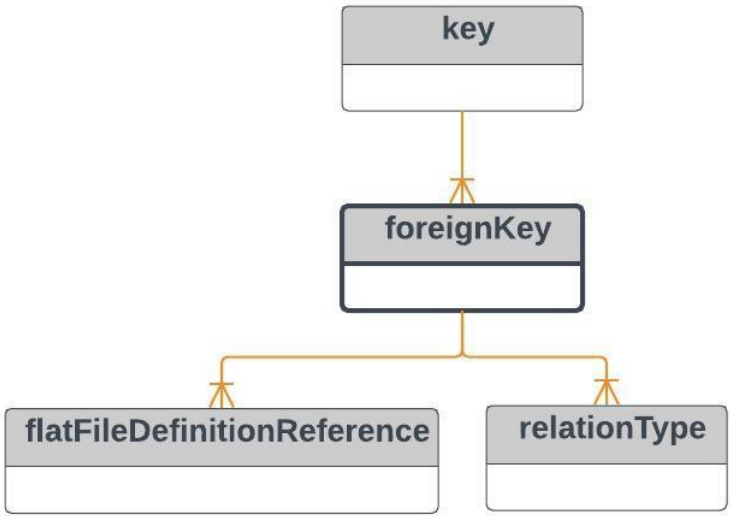
Underliggende elementer	Kravtype	Forekomster
flatFileType	mandatory	1 - n

Eksempel:

```
<structureTypes>
   <flatFileTypes>
      <flatFileType name=”flatfil1”>
         ….
      </flatFileType>
      <flatFileType name=”flatfil2”>
         ….
      </flatFileType>
      ….
   </flatFileTypes>
   ….
</structureTypes>
```

Eksempelet viser en flatFileTypes hvor det er definert to typer – flatfil1 og flatfil2, og med en indikasjon på flere.

foreignKey



Elementnavn	Kravtype	Beskrivelse
foreignKey	optional	Dette elementet med underelementer benyttes for å angi fremmednøkler.

Ingen attributter.

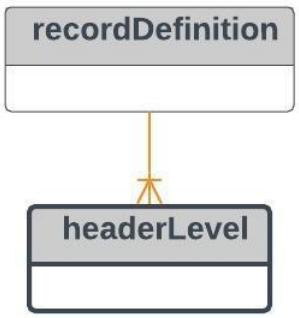
Underliggende elementer	Kravtype	Forekomster
flatFileDefinitionReference	mandatory	1
relationType	mandatory	1

Eksempel:

```
<key name=”saknr1”>
   <foreignKey>
      <flatFileDefinitionReference name=”sak”>
         <recordDefinitionReferences>
            <recordDefinitionReference name=”post1”>
               <fieldDefinitionReferences>
                  <fieldDefinitionReference name=”saknr”/>
               </fieldDefinitionReferences>
            </recordDefinitionReference>
         </recordDefinitionReferences>
      </flatFileDefinitionReference>
      <relationType type=”1:n”/>
   </foreignKey>
   <fieldDefinitionReferences>
      <fieldDefinitionReference name=”saksnr”/>
   </fieldDefinitionReferences>
</key>
```

Eksempelet viser en fremmednøkkel ved navn saknr1. I fildefinisjonen nøkkelen er definert innen er det også et felt ved navn saksnr definert. Dette er det interne feltet i nøkkeldefinisjonen. Nøkkelen viser så over til en annen fil ved navn sak med underliggende post med navn post1 hvor det er definert et felt med navn saknr. Relasjonen er også satt til å være en til mange.

headerLevel



Elementnavn	Kravtype	Beskrivelse
headerLevel	optional	Dette elementet benyttes for å angi om det finnes overskriftslinjer og antallet av disse. Dersom elementet ikke er oppgitt, vil det være å forstå som at det ikke finnes overskriftslinjer.

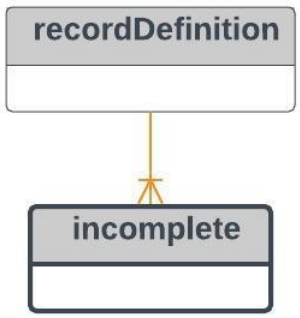
Ingen attributter og ingen underliggende elementer.

Eksempel:

<recordDefinition name=”**post1**” type=”**posttype1**”>
  <fieldDefinitions>
    ….
  </fieldDefinitions>
  <headerLevel type=”**2**”>
</recordDefinition>

Eksempelet viser at det for poster som følger post1 er av posttype1 og at de to første postene er overskrifter.

incomplete



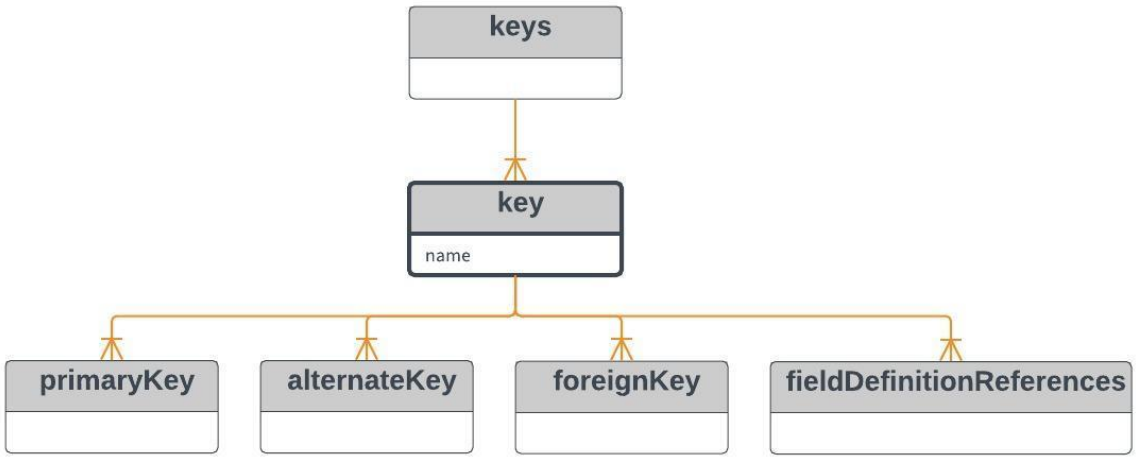
Elementnavn	Kravtype	Beskrivelse
incomplete	optional	Elementet benyttes for å angi at en post ikke er definert i sin helhet. Benyttes når man bare oppgir enkelte felter (elementer) i en post. Se også *external*.

Ingen attributter og ingen underliggende elementer.

Elementet incomplete henger sammen med elementet external.

For forklaring og eksempel, se external.

key



Elementnavn	Kravtype	Beskrivelse
key	mandatory	Som med de andre elementene er også nøkler definert med et samlenivå og et detaljnivå. For hver nøkkel vil det angis hva slags type nøkkel det er – primærnøkkel, sekundærnøkkel eller fremmednøkkel.

Attributter	Kravtype	Beskrivelse
name	mandatory	

Underliggende elementer	Kravtype	Forekomster
primaryKey	optional	0 - 1
alternateKey	optional	0 - 1
foreignKey	optional	0 - 1
fieldDefinitionReferences	mandatory	1

NB!	Når det gjelder de underliggende elementene «primaryKey», «alternateKey» og «foreignKey» er de i utgangspunktet optional. Dog må en og kun en av de være tilstede.

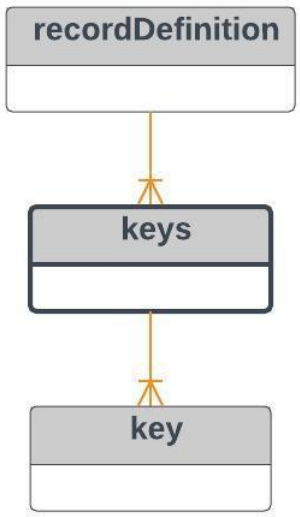
Eksempel:

```
<key name=”primær”>
   <primaryKey/>
   <fieldDefinitionReferences>
      <fieldDefinitionReference name=”saksnr”/>
   </fieldDefinitionReferences>
</key>


<key name=”alternativ”>
   <alternateKey/>
   <fieldDefinitionReferences>
      <fieldDefinitionReference name=”saksnr”/>
   </fieldDefinitionReferences>
</key>


<key name=”saknr1”>
   <foreignKey>
      <flatFileDefinitionReference name=”sak”>
         <recordDefinitionReferences>
            <recordDefinitionReference name=”post1”>
               <fieldDefinitionReferences>
                  <fieldDefinitionReference name=”saknr”/>
               </fieldDefinitionReferences>
            </recordDefinitionReference>
         </recordDefinitionReferences>
      </flatFileDefinitionReference>
      <relationType type=”1:n”/>
   </foreignKey>
   <fieldDefinitionReferences>
      <fieldDefinitionReference name=”saksnr”/>
   </fieldDefinitionReferences>
</key>
```

I eksemplene er definert en nøkkel – navn primær for en primærnøkkel, van alternativ for en alternativnøkkel og navn saknr1 for en fremmednøkkel, som alle er basert på elementet saksnr som selve nøkkelfeltet. (Dette kan godt bestå av mer enn ett element.)

For primærnøkkel og alternativnøkkel er dette det hele, da disse nøklene gjelder for den aktuelle posten elementene er definert inn i. For fremmednøkkel må det også angis et tilsvarende element (evt. elementer) i en annen post, i dette tilfellet til elementet saknr i posten post1 i filen sak.

keys



Elementnavn	Kravtype	Beskrivelse
keys	optional	Som med de andre elementene er også nøkler definert med et samlenivå og et detaljnivå. For hver nøkkel vil det angis hva slags type nøkkel det er – primærnøkkel, sekundærnøkkel eller fremmednøkkel.

Ingen attributter.

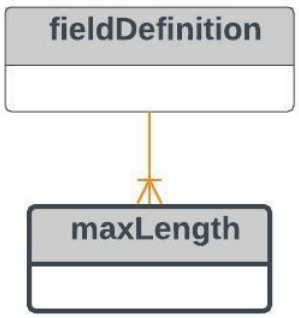
Underliggende elementer	Kravtype	Forekomster
key	mandatory	1 - n

Eksempel:

```
<recordDefinition name=”post1”>
   …
   <keys>
      <key name=”primær”>
         …
      <key>
      <key name=”post1”>
         …
      </key>
   </keys>
   <fieldDefinitions>
      …
   </fieldDefinitions>
</recordDefinition>
```

I eksempelet er angitt en posttype ved navn post1. For denne posttypen er det angitt feltdefinisjoner gjennom fieldDefinitions og et sett av nøkler gjennom keys. Av nøkler er det angitt to ved navn primær og post1 respektive.

maxLength



Elementnavn	Kravtype	Beskrivelse
maxLength	optional	Dette elementet benyttes for å angi maksimumslengden av et felt (element).

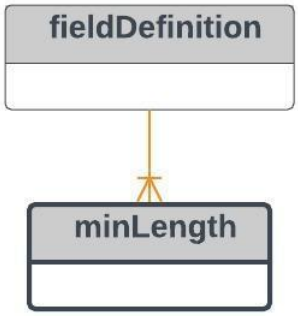
Ingen attributter og ingen underliggende elementer.

Eksempel:

<fieldDefinition name=”**name**” typeReference=”**field1**”>
….
<maxLength>***30***</maxLength>
….
</fieldDefinition>

I eksemplet er angitt et felt ved navn name med maksimum lengde på 30 tegn.

minLength



Elementnavn	Kravtype	Beskrivelse
minLength	optional	Dette elementet benyttes for å angi minimumslengden av et felt (element).

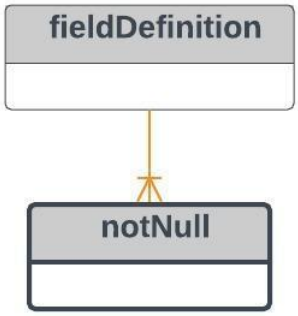
Ingen attributter og ingen underliggende elementer.

Eksempel:

<fieldDefinition name=”**name**” typeReference=”**field1**”>
….
<minLength>***30***</minLength>
….
</fieldDefinition>

I eksemplet er angitt et felt ved navn name med minimum lengde på 30 tegn.

notNull



Elementnavn	Kravtype	Beskrivelse
notNull	optional	Dette elementet benyttes til å angi om et felt (element) kan inneholde NULL-verdi eller ikke.

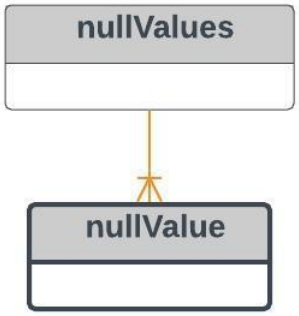
Ingen attributter og ingen underliggende elementer.

Eksempel:

<fieldDefinition name=”**name**” typeReference=”**field1**”>
….
<notNull/>
….
</fieldDefinition>

I eksemplet er angitt et felt ved navn name hvor verdien ikke kan være NULL.

nullValue

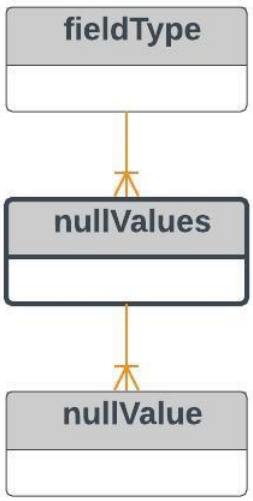


Elementnavn	Kravtype	Beskrivelse
nullValue	mandatory	Elementet benyttes for å angi en verdi som skal tolkes som NULL-verdi i et felt (element).

Ingen attributter og ingen underliggende elementer.

Se eksempel og forklaring under nullValues.

nullValues



Elementnavn	Kravtype	Beskrivelse
nullValues	optional	Elementet er et samleelement for gyldige NULL-verdier i et felt (element).

Ingen attributter.

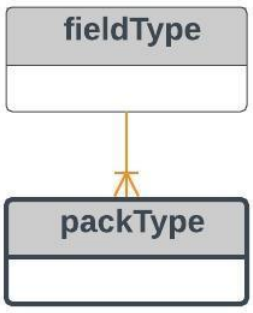
Underliggende elementer	Kravtype	Forekomster
nullValue	mandatory	1 - n

Eksempel:

```
<fieldTypes>
   ….
   <fieldType name=”dato8”>
      <dataType>date</dataType>
      <fieldFormat>YYYY.MM.DD</fieldFormat>
      <nullValues>
         <nullValue>9999.12.31</nullValue>
      </nullValues>
   </fieldType>
   ….
</fieldTypes>
```

I eksemplet er definert en felttype ved navn dato8. Felttypen angir at et felt av denne typen er av dataformat dato (date) og at datoen skal være på formen YYYY.MM.DD som f.eks. 2018.06.28. I tillegg er 9999.12.31 definert som NULL-verdi. Dette for at datoen skal være 0gyldig i henhold til formatet.

packType



Elementnavn	Kravtype	Beskrivelse
packType	optional	Dette elementet benyttes for å angi at feltet (elementet) er pakket. Som eksemplet under viser, er det opp til brukeren å definere på hvilket sett man vil angi pakking.

Ingen attributter og ingen underliggende elementer.

Eksempel:

<fieldType name=”**streng**”>
....
<packType>***packed***</packType>
....
</fieldType>

I eksemplet over er elementet packType benyttet som et flagg hvor det er angitt bare med verdien ”packed” at verdien i elementet er pakket.

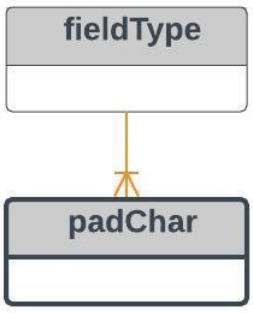
Denne måten å bruke elementet på kan også gjøres på denne måten:

<packType/>

Hvor bare eksistensen av elementet er nok til å angi at verdien er pakket.

Alternativt kan man også angi navnet på pakkealgoritmen som verdi i elementet.

padChar



Elementnavn	Kravtype	Beskrivelse
padChar	optional	Dette elementet benyttes for å angi hvilket tegn som er benyttet for padding, dvs «fylle ut» et felt (element).

Ingen attributter og ingen underliggende elementer.

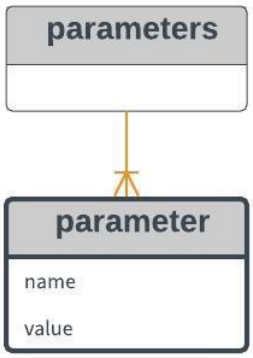
Eksempel:

<fieldType name=”**streng**”>
....
<padChar> </padChar>
....
</fieldType>

I eksemplet over er det angitt at verdien i elementet er paddet ut til full lengde med blanke tegn som fyllingstegn.

OBS! Vær oppmerksom på at fyllingstegn både kan forekomme før den reelle verdien og etter. I eksemplet er det ikke angitt ”alignment”, hvilket innebærer at ”standardverdien” av justering som vil være venstrestilt og dermed fyllingstegn etter verdien. Hadde ”alignment” angitt høyrejustering, ville det ført fyllingstegn før den reelle verdien.

parameter



Elementnavn	Kravtype	Beskrivelse
parameter	optional	Elementet benyttes for å angi en parameter til en prosess.

Attributter	Kravtype	Beskrivelse
name	mandatory	
value	optional	

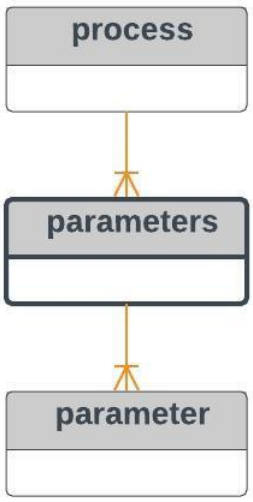
Ingen underliggende elementer.

Eksempel:

<parameters>
  <parameter name=”**tlf**” value=”**12345678**”/>
</parameters>

I eksemplet over er det definer en parameter ved navn ”tlf” med verdien ”12345678”. Det er kun definert denne ene parameteren.

parameters



Elementnavn	Kravtype	Beskrivelse
parameters	optional	Elementet er et samleelement for parametre til en prosess.

Ingen attributter.

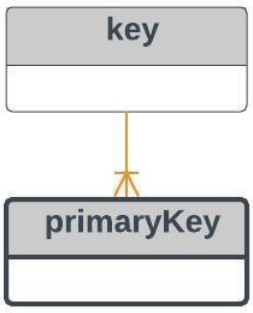
Underliggende elementer	Kravtype	Forekomster
parameter	mandatory	1 - n

Eksempel:

<process name=”**fnr**”>
  <parameters>
    <parameter name=”**fdato**” value=”**123456**”/>
    <parameter name=”**pnr**” value=”**12345**”/>
  </parameters>
</process>

I eksemplet over er oppgitt en prosess som har to parameter. Prosessen heter ”fnr” og parameterne heter ”fdato” og ”pnr”. Parameterne har verdiene ”123456” og ”12345” respektive. Dette kunne være et eksempel på hvordan man identifiserer objekter knyttet til et fødselsnr i et arkiv.

primaryKey



Elementnavn	Kravtype	Beskrivelse
primaryKey	optional	Dette elementet benyttes for å angi om en nøkkel er en primærnøkkel eller ikke.

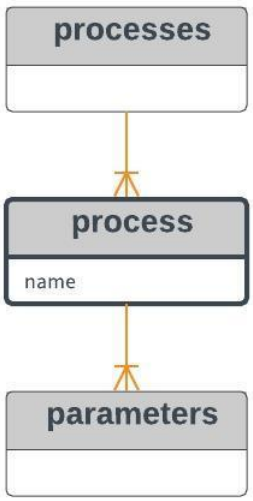
Ingen attributter og ingen underliggende elementer.

Eksempel:

```
<key name="indeks1">
   <primaryKey/>
   <fieldDefinitionReferences>
      ....
   </fieldDefinitionReferences>
</key>
```

I eksemplet er det definert en nøkkel indeks1 som en primærnøkkel (evt en indeks) og hvor de elementene som inngår er definert under *fieldDefinitionReferences* (her bare angitt med ….).

process



Elementnavn	Kravtype	Beskrivelse
process	mandatory	Det kan defineres prosesser i en ADDML-fil. Med prosesser menes operasjoner som en eller annet applikasjon skal utføre. Om det er ønskelig kan man benytte prosesser på en annen måte. Elementet *processes* er samlenivået, mens *process* er detaljnivået.

Attributter	Kravtype	Beskrivelse
name	mandatory	

Underliggende elementer	Kravtype	Forekomster
Parameters	optional	0 - 1

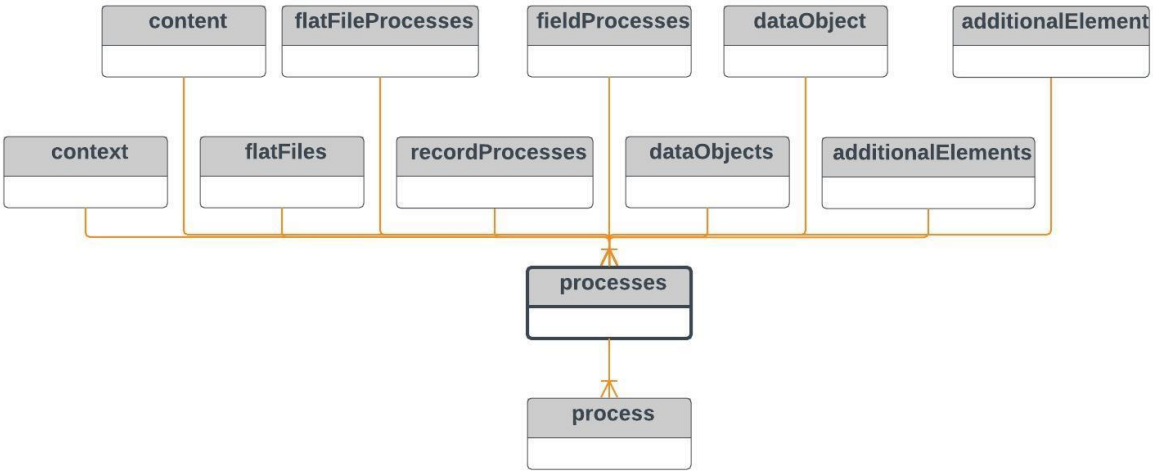
Eksempel:

<processes>
  <process name=”**test1**”/>
  <process name=”**test2**”/>
  …
</processes>

I eksemplet er angitt et sett av prosesser som skal utføres av en applikasjon. Elementet ”process” er i utgangspunktet bare et flagg som anger at en prosess skal utføres. Hva slags aktivitet som faktisk skal utføres blir ikke definert i standarden ADDML, men er noe som brukerne og leverandørene av applikasjonene som benyttes må være enige om.

Bruken av prosesser kan være av forskjellig art. I noen sammenhenger kan prosesser benyttes for å angi tester som skal utføres. I andre sammenhenger kan prosesser benyttes for identifisering av materiale som tilfredstiller visse kriterier. Man kan også tenke seg å benytte prosesser i sammenheng med vedlikehold av materiale, f.eks. ved å angi kriterier for konvertering. Mulighetene er mange, men forutsetter altså at det er laget software som kan utføre den/de ønskede aktivitetene.

processes



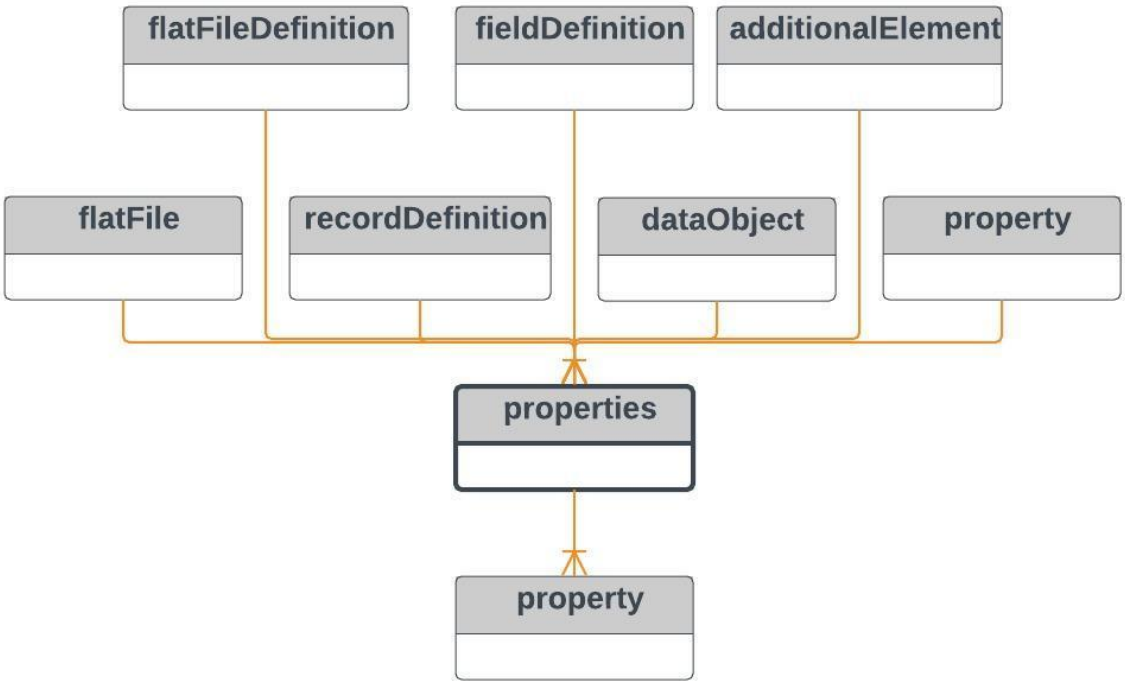
Elementnavn	Kravtype	Beskrivelse
processes	optional	Det kan defineres prosesser i en ADDML-fil. Med prosesser menes operasjoner som en eller annet applikasjon skal utføre. Om det er ønskelig kan man benytte prosesser på en annen måte. Elementet *processes* er samlenivået, mens *process* er detaljnivået.

Ingen attributter.

Underliggende elementer	Kravtype	Forekomster
process	mandatory	1 - n

For eksempel og kommentar se under *process*.

properties



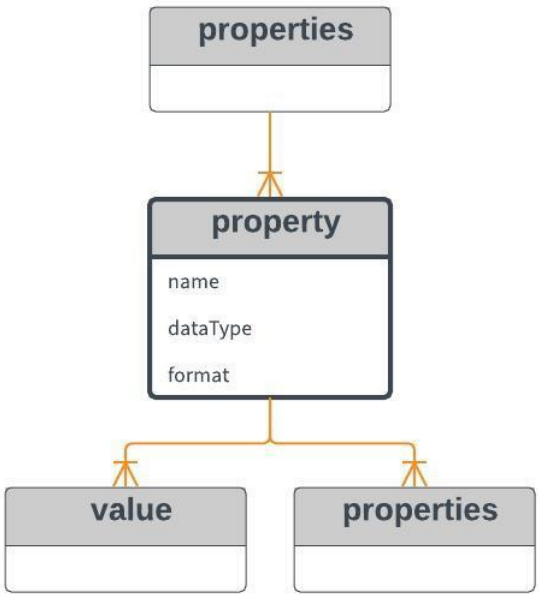
Elementnavn	Kravtype	Beskrivelse
properties	optional	For mange elementer i ADDML kan man også bygge opp et hierarki av egenskaper knyttet til elementet. Elementet *properties* er samlelemenetet for en gruppe egenskaper, mens elementet *property* er elementet for den enkelte egenskap.

Ingen attributter.

Underliggende elementer	Kravtype	Forekomster
property	mandatory	1 - n

For eksempel se *property*.

property



Elementnavn	Kravtype	Beskrivelse
property	mandatory	For mange elementer i ADDML kan man også bygge opp et hierarki av egenskaper knyttet til elementet. Elementet *properties* er samlelemenetet for en gruppe egenskaper, mens elementet *property* er elementet for den enkelte egenskap.

Attributter	Kravtype	Beskrivelse
name	mandatory	
dataType	optional	
Format	optional	

Underliggende elementer	Kravtype	Forekomster
value	optional	0 - 1
properties	optional	0 - 1

Eksempel:

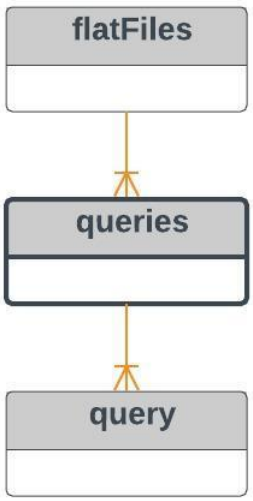
```
<properties>
   <property name=”email”>
      <value>addml@arkivverket.no</value>
   </property>
</properties>
```

I eksemplet er det angitt en egenskap ved navn ”email” med verdien ”addml@arkivverket.no”. Eksemplet viser hvordan man enkelt kan definere en egenskap. De neste to eksemplene viser hvordan det er mulig å bygge opp et hierarki av egenskaper.

```
<properties>
   <property name=”type”>
      <value>Noark 5</value>
      <properties>
         <property name=”version”>
            <value>5.4</value>
         </property>
      </properties>
</properties>
```

```
<properties>
   <property name=”comments”>
      <properties>
         <property name=”comment”>
            <value>Kommentar 1</value>
         <property>
         <property name=”comment”>
            <value>Kommentar 2</value>
         <property>
      <properties>
   <property>
</properties>
```

queries



Elementnavn	Kravtype	Beskrivelse
queries	optional	Tanken med disse elementene er å kunne beskrive spørringene mot datafilene, f.eks. i form av SQL-setninger. Slike spørringer kan dokumentere hvordan informasjonen har vært brukt og hvordan selve uttrekket har foregått. I tillegg vil det være mulighet for andre applikasjoner på et senere tidspunkt å kunne nyttiggjøre seg disse spørringene.

Ingen attributter.

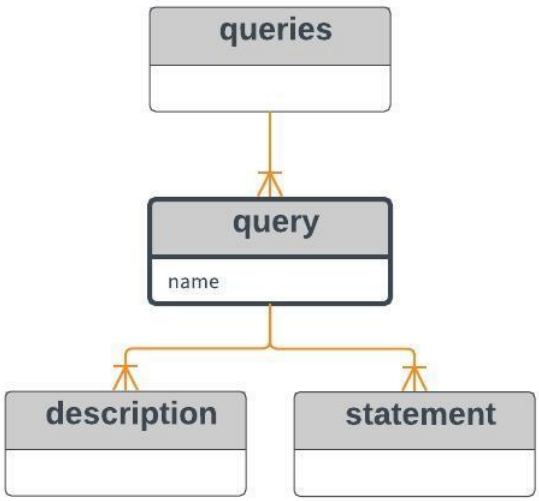
Underliggende elementer	Kravtype	Forekomster
query	mandatory	1 - n

Eksempel:

```
<flatFiles>
   …
   <queries>
      <query name=”sok1”>
         <description>Beskrivelse av søket</description>
         <statement>Selve query’en - programkoden</statement>
      </query>
      …
   </queries>
  …
</flatFiles>
```

Eksemplet viser hvordan queries kan knyttes opp mot elementet *flatFiles*.

query



Elementnavn	Kravtype	Beskrivelse
query	mandatory	Tanken med disse elementene er å kunne beskrive spørringene mot datafilene, f.eks. i form av SQL-setninger. Slike spørringer kan dokumentere hvordan informasjonen har vært brukt og hvordan selve uttrekket har foregått. I tillegg vil det være mulighet for andre applikasjoner på et senere tidspunkt å kunne nyttiggjøre seg disse spørringene.

Attributter	Kravtype	Beskrivelse
name	mandatory	

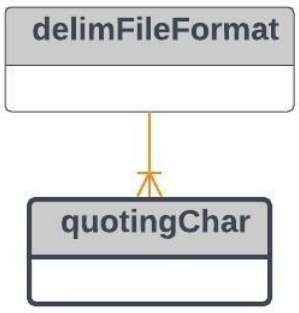
Underliggende elementer	Kravtype	Forekomster
description	optional	0 - 1
statement	mandatory	1

Eksempel:

```
<queries>
   <query name=”sok1”>
      <description>Beskrivelse av søket</description>
      <statement>Selve query’en - programkoden</statement>
   </query>
   …
</queries>
```

I eksemplet definers et søk hvor søket er angitt med navn ”sok1” og med selve programkoden – normalt en sql-kode, og en beskrivelse som bør fortelle hva koden er ment å gi.

quotingChar



Elementnavn	Kravtype	Beskrivelse
quotingChar	optional	Dette elementet benyttes for å angi hvilke(t) tegn som benyttes som innkapslingstegn av en tekststreng i en tegnseparert fil.

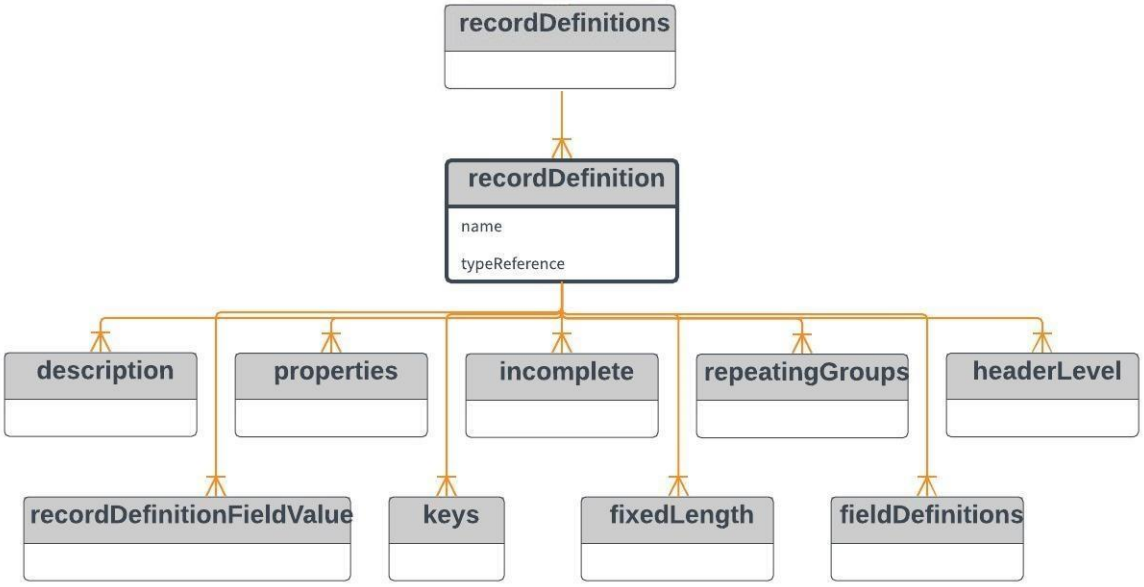
Ingen attributter og ingen underliggende elementer.

Eksempel:

<flatFileType name=”**fil1**”>
….
<delimFileFormat>
<fieldSeparatingChar>**;**<fieldSeparatingChar>
<quotingChar>**”**</quotingChar>
</delimFileFormat>
</flatFileType>

I eksemplet er det vist hvordan man kan angi ; (semikolon) som skilletegn mellom de enkelte feltene og ” (dobbel apostrof) som innkapslingstegn for tekst.

recordDefinition



Elementnavn	Kravtype	Beskrivelse
recordDefinition	mandatory	I likhet med som for filer, er det også en tilsvarende struktur på definisjonsnivået for poster. Det er en oppdeling av definisjonsnivået i et samleelement og et definert element for hver post. Disse elementene er *recordDefinitions* og *recordDefinition* respektive. Fra *recordDefinition* finnes det en referanse til *recordType*.

Attributter	Kravtype	Beskrivelse
name	mandatory	
typeReference	optional	

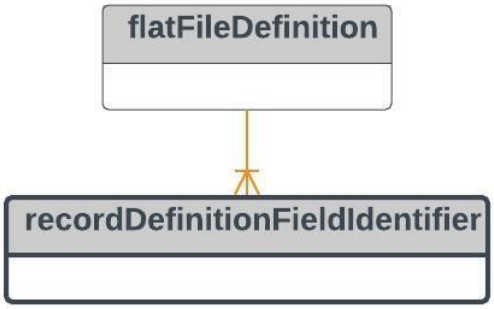
Underliggende elementer	Kravtype	Forekomster
description	optional	0 - 1
properties	optional	0 - 1
recordDefinitionFieldValue	optional	0 - 1
incomplete	optional	0 - 1
fixedLength	optional	0 - 1
repeatingGroups	optional	0 - 1
keys	optional	0 - 1
fieldDefinitions	mandatory	1
headerLevel	optional	0 - 1

Eksempel:

```
<recordDefinitions>
   …
   <recordDefinition name=”Posttype sak” typeReference=”rectyp 1”>
      <description>En beskrivelse av posttypen.</description>
      <properties>
         <property name=”Egenskap 1”>
            …
         </property>
         …
      </properties>
      <recordDefinitionFieldValue>S</recordDefinitionFieldValue>
      <fixedLength>280</fixedLength>
      <keys>
         <key name=”Primær”>
            …
         </key>
         …
      </keys>
      <fieldDefinitions>
         <fieldDefinition name=”Type” typeReference=”fieldtyp 1”>
            …
         </fieldDefinition>
         …
      </fieldDefinition>
   </recordDefinition>
   …
</recordDefinitions>
```

I eksemplet er det definert en posttype ved navn “Posttype sak”, som er en posttype av typen “rectyp 1”. Posttypen har flere egenskaper, hvor den første ved navn “Egenskap 1” er angitt, mens de resterende bare er markert med tre prikker. Posttypen er en av flere posttyper hvorav denne er identifisert med verdien S. (Feltet det er snakk om er definert i overliggende fileDefinition og dermed ikke angitt i dette eksemplet.) Posttypen har fast lengde på 280 tegn og har også definert noen nøkler og noen underliggende felter.

recordDefinitionFieldIdentifier

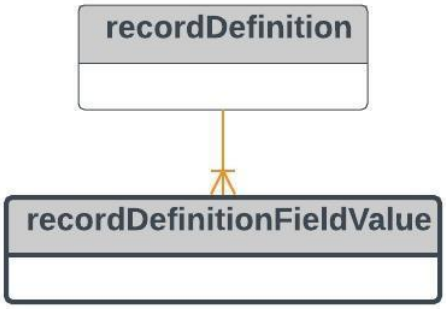


Elementnavn	Kravtype	Beskrivelse
recordDefinitionFieldIdentifier	optional	Dette elementet benyttes for å angi hvilket felt (element) som definerer hvilken posttype en post i en fil er.

Ingen attributter og ingen underliggende elementer.

For eksempel se eksemplet under recordDefinitionFieldValue.

recordDefinitionFieldValue



Elementnavn	Kravtype	Beskrivelse
recordDefinitionFieldValue	optional	Dette elementet benyttes for å angi verdien som bestemmer hvilken posttype denne posten er.

Ingen attributter og ingen underliggende elementer.

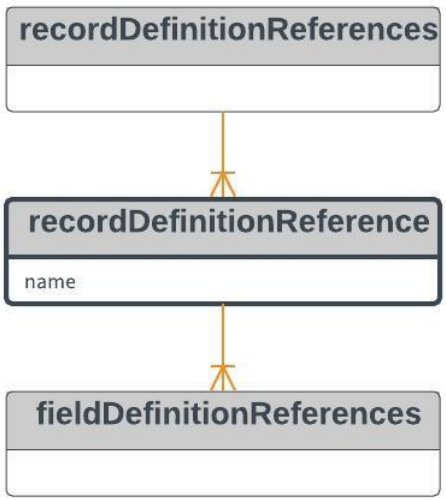
Eksempel:

```
<flatFileDefinition name="Noark3" typeReference="Noark3">
   …
   <recordDefinitionFieldIdentifier>Type</recordDefinitionFieldIdentifier>
   <recordDefinitions>
      <recordDefinition name="Sak" typeReference="File">
         …
         <recordDefinitionFieldValue>S</recordDefinitionFieldValue>
         …
      </recordDefinition>
      <recordDefinition name="Dokument" typeReference="Document">
         …
         <recordDefinitionFieldValue>D</recordDefinitionFieldValue>
         …
      </recordDefinition>
      …
   </recordDefinitions>
</flatFileDefinition>
```

I eksemplet er det vist hvordan man bruker elementene recordDefinitionFieldIdentifier og recordDefinitionFieldValue dersom en fil inneholder flere forskjellige posttyper. Eksemplet viser at feltet som benyttes heter «Type» og at poster som er av typen Sak har verdien S i elementet, mens poster av typen Dokument har verdien D.

OBS! Standarden sier ingenting om hvordan man angir elementet som inneholder verdiene. I eksemplet er det angitt med navn på elementet, men man kan også like gjerne benytte et nummer som angir hvilket element i feltrekkefølgen som skal benyttes.

recordDefinitionReference



Elementnavn	Kravtype	Beskrivelse
recordDefinitionReference	mandatory	Dette elementet benyttes for å gi en referanse til posten for en fremmednøkkel.

Attributter	Kravtype	Beskrivelse
name	mandatory	

Underliggende elementer	Kravtype	Forekomster
fieldDefinitionReferences	optional	0 - 1

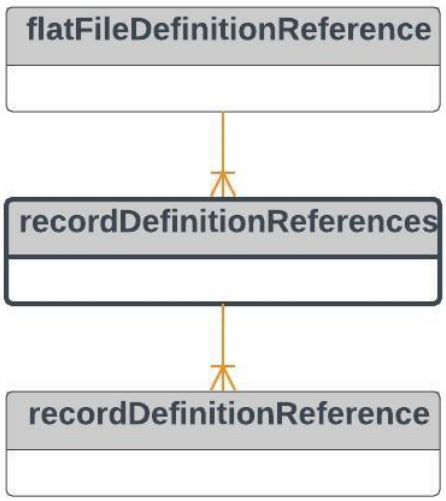
Eksempel:

```
<recordDefinitionReferences>
   <recordDefinitionReference name=”post2”>
      <fieldDefinitionReferences>
         <fieldDefinitionReference name=”postnr”/>
      </fieldDefinitionReferences>
   </recordDefinitionReference>
</recordDefinitionReferences>
```

Eksemplet viser en referanse til feltet postnr i posten post2.

Dette elementet benyttes for henvisning til fremmednøkler.

recordDefinitionReferences



Elementnavn	Kravtype	Beskrivelse
recordDefinitionReferences	optional	Dette elementet er et samleelement for en postreferanse.

Ingen atttributter.

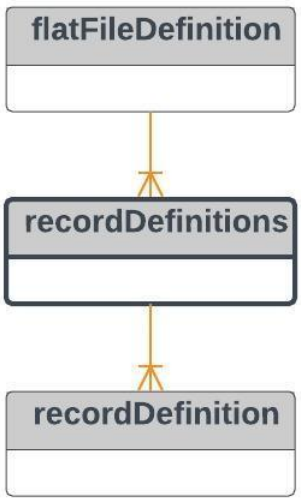
Underliggende elementer	Kravtype	Forekomster
recordDefinitionReference	mandatory	1 - n

Eksempel:

```
<foreignKey>
   <flatFileDefinitionReference name=”noarksak”>
      <recordDefinitionReferences>
         <recordDefinitionReference name=”post2”>
            <fieldDefinitionReferences>
               <fieldDefinitionReference name=”postnr”/>
            </fieldDefinitionReferences>
         </recordDefinitionReference>
      </recordDefinitionReferences>
   </flatFileDefinitionReference>
   <relationType>1:n</relationType>
</foreignKey>
```

Eksemplet viser hvordan dette elementet inngår som en del av referansen for en fremmednøkkel.

recordDefinitions



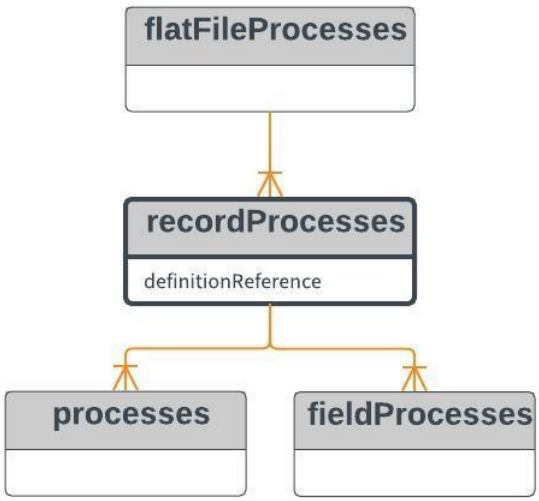
Elementnavn	Kravtype	Beskrivelse
recordDefinitions	mandatory	I likhet med som for filer, er det også en tilsvarende struktur på definisjonsnivået for poster. Det er en oppdeling av definisjonsnivået i et samleelement og et definert element for hver post. Disse elementene er *recordDefinitions* og *recordDefinition* respektive. Fra *recordDefinition* finnes det en referanse til *recordType*.

Ingen attibutter.

Underliggende elementer	Kravtype	Forekomster
recordDefinition	mandatory	1 - n

For eksempel se under elementene *external* og *recordDefinition*.

recordProcesses



Elementnavn	Kravtype	Beskrivelse
recordProcesses	optional	Som for felter (elementer) og filer, er det også mulig å angi prosesser på poster. Dette elementet benyttes for dette formålet.

Attributter	Kravtype	Beskrivelse
name	mandatory	

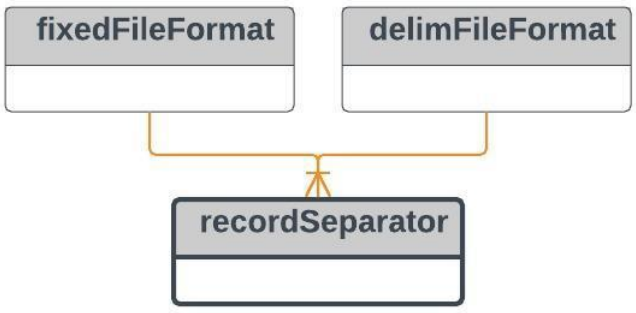
Underliggende elementer	Kravtype	Forekomster
processes	optional	0 - 1
fieldProcesses	optional	0 - n

Eksempel:

*<flatFileProcesses flatFileReference=”***fildef1***”>*
*<recordProcesses definitionReference=”***postdef1***”>*
*<fieldProcesses definitionReference=”***yrke***”>*
<processes>
*<process name=”***Control_Codes***”/>*
</processes>
</fieldProcesses>
</recordProcesses>
</flatFileProcesses>

I eksemplet er det definert en prosess som heter Control_Codes på elementet yrke i posttypen postdef1.

recordSeparator



Elementnavn	Kravtype	Beskrivelse
recordSeparator	optional	I dette elementet angis hva slags tegn som er benyttet som skilletegn mellom poster i en fil.

Ingen attributter og ingen underliggende elementer.

NB!	Elementet «recordSeparator» er valgfri med «fixedFileFormat» som overliggende element. Dersom «delimFileFormat» er overliggende element er elementet «recordSeparator» obligatorisk.

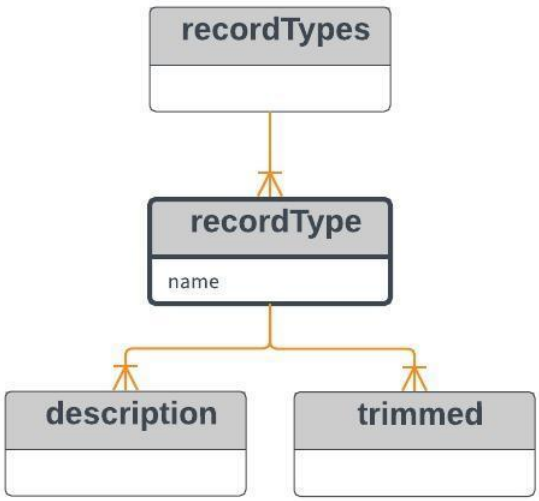
Eksempel:

```
<flatFileType name="fil1">
   ….
   <fixedFileFormat>
      <recordSeparator>CRLF</recordSeparator>
   </fixedFileFormat>
</flatFileType>
```

I eksemplet er angitt en flat fil ved navn fil1 som har CRLF (Carriage Linefeed, dvs vanlig ny linje) for hver av portene i filen.

OBS! Viktig å være oppmerksom på at dersom *recordSeparator* ikke er angitt, vil det bety at det ikke er noe separasjonstegn mellom postene.

recordType



Elementnavn	Kravtype	Beskrivelse
recordType	mandatory	For typene på postnivå benyttes elementene *recordTypes* og *recordType*, igjen med et samlenivå og et detaljnivå. På postnivå kan det angis et navn på typen, en beskrivelse (hvis man ønsker), samt om feltene er trimmet, altså om ledende nuller og etterfølgende blanke er fjernet i feltene.

Attributter	Kravtype	Beskrivelse
name	mandatory	

Underliggende elementer	Kravtype	Forekomster
description	optional	0 - 1
trimmed	optional	0 - 1

Eksempel:

<recordTypes>
*<recordType name=”***typepostdef1***”>*
<trimmed/>
</recordType>
</recordTypes>

Eksemplet viser definisjonen av en posttype ved navn typepostdef1. I denne typen er angitt at posten er trimmet.

recordTypes

Elementnavn	Kravtype	Beskrivelse
recordTypes	optional	For typene på postnivå benyttes elementene *recordTypes* og *recordType*, igjen med et samlenivå og et detaljnivå. På postnivå kan det angis et navn på typen, en beskrivelse (hvis man ønsker), samt om feltene er trimmet, altså om ledende nuller og etterfølgende blanke er fjernet i feltene.

Ingen attributter.

Underliggende elementer	Kravtype	Forekomster
recordType	mandatory	1 - n

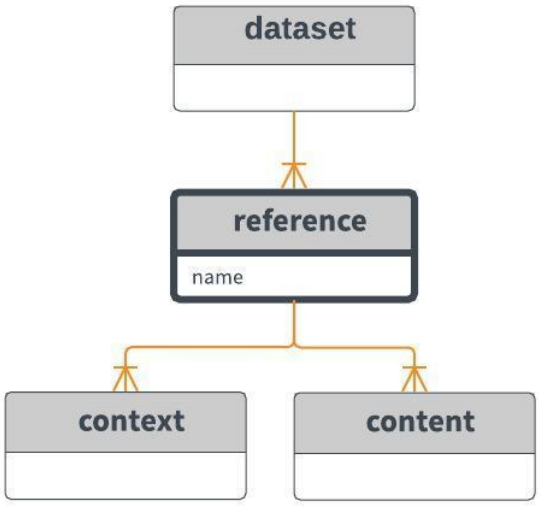
Eksempel:

```
<structureTypes>
  <flatFileTypes>
    <flatFileType name="typefildef1">
      ...
    </flatFileType>
  </flatFileTypes>
  <recordTypes>
    <recordType name="typepostdef1"/>
  </recordTypes>
  <fieldTypes>
    <fieldType name="typefeltdef1">
      ...
    </fieldType>
  </fieldTypes>
</structureTypes>
```

Eksemplet viser hvordan man grupper definisjonene av filtyper, posttyper og felttyper under elementet *structtureTypes*. I eksemplet er kun angitt en enkelt type av hver. Det er fullt tillatt å ha flere av samme type. Se eksempel under *structureTypes*.

reference

Elementet *reference* med underliggende elementer og attributter inneholder metadata om datasettet. Elementet i seg selv er bare et samlenivå.



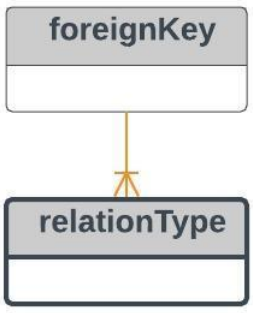
Elementnavn	Kravtype	Beskrivelse
reference	optional	*reference* er et samlenivå for administrative informasjon om datasettet.

Attributter	Kravtype	Beskrivelse
name	optional	*name* angir navnet til det spesifikke datasettet. Det er ingen krav til navnsetting i utgangspunktet, men navnene for dataset må være unike innen en addml-fil.

Underliggende elementer	Kravtype	Forekomster
context	optional	0 - 1
content	optional	0 - 1

For eksempel se innledningen under hovedkapitlet «Beskrivelse av ADDML» og første delkapittel «Hva beskrives i ADDML 8.3. (Hovedstruktur)» og andre delkapittel «Tilhørende informasjon – bevaringsverdige metadata».

relationType



Elementnavn	Kravtype	Beskrivelse
relationType	mandatory	Angivelse av forholdet i en relasjon i en fremmednøkkel.

Ingen attributter og ingen underliggende elementer.

Eksempel:

```
<foreignKey>
   <flatFileDefinitionReference name=”fil2”>
      <recordDefinitionReferences>
         <recordDefinitionReference name=”post2”>
            <fieldDefinitionReferences>
               <fieldDefinitionReference name=”postnr”/>
            </fieldDefinitionReferences>
         </recordDefinitionReference>
      </recordDefinitionReferences>
   </flatFileDefinitionReference>
   <relationType>n:1</relationType>
</foreignKey>
<fieldDefinitionReferences>
   <fieldDefinitionReference name=”postnr”>
</fieldDefinitionReferences>
```

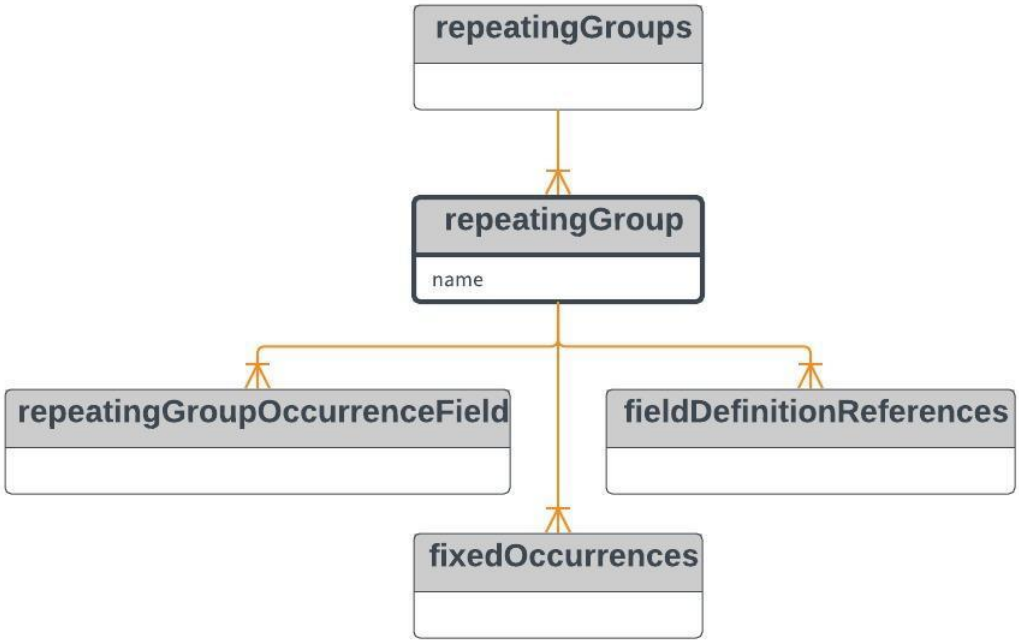
Eksemplet viser at forholdet mellom fremmedverdien i elementet postnr i posten post2 i filen fil2 mot elementet postnr i den interne posten er mange til en (n:1).

ADDML setter ingen krav til hvordan forholdet skal angis, men normalt vil følgende verdier være aktuelle:

1:1, n:1, n:n

(Dog vil siste verdi gjøre ting vanskelig å tolke informasjonen i et arkivuttrekk.)

repeatingGroup



Elementnavn	Kravtype	Beskrivelse
repeatingGroup	mandatory	Dette elementet benyttes for å angi repeterende grupper av felt (elementer). Elementet *repeatingGroups* som et overordnet samleelement og dette elementet som detaljnivået.

Attributter	Kravtype	Beskrivelse
name	mandatory	

Underliggende elementer	Kravtype	Forekomster
repeatingGroupOccurrenceField	optional	0 - 1
fixedOccurrences	optional	0 - 1
fieldDefinitionReferences	mandatory	1

NB!	Når det gjelder de underliggende elementene «repeatingGroupOccurrenceField» og «fixedOccurrences» er de optional. Dog kan kun en av de være tilstede samtidig.

Eksempel 1:

```
<repeatingGroups>
   <repeatingGroup name=”kontingent”>
      <fixedOccurrences>5</fixedOccurrences>
      <fieldDefinitionReferences>
         ….
      </fieldDefinitionReferences>
   </repeatingGroup>
</repeatingGroups>
```

I eksemplet er angitt en repeterende gruppe som består av flere elementer, kalt kontingent, som gjentas som en gruppe 5 ganger.

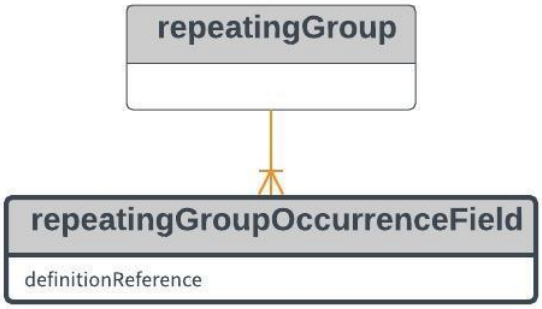
Eksempel 2:

```
<repeatingGroups>
   <repeatingGroup name=”kontingent”>
      <repeatingGroupOccurrenceField definitionReference=”postnr”/>
   </repeatingGroup>
</repeatingGroups>
```

I eksemplet er angitt en repeterende gruppe kalt kontingent med ett enkelt element («postnr»).

Siden denne varianten ikke angir antall repetisjoner, kan den kun benyttes når den repeterende gruppen er de siste elementene i posten.

repeatingGroupOccurrenceField



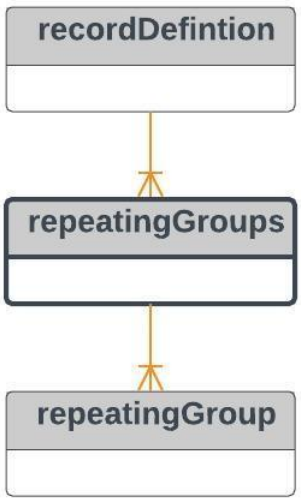
Elementnavn	Kravtype	Beskrivelse
repeatingGroupOccurrenceField	optional	Dette elementet benyttes til å angi hvor ganger en repeterende gruppe forekommer.

Attributter	Kravtype	Beskrivelse
definitionReference	mandatory	

Ingen underliggende elementer.

For eksempel se eksempel 2 under elementet *repeatingGroup*.

repeatingGroups



Elementnavn	Kravtype	Beskrivelse
repeatingGroups	optional	Dette elementet benyttes for å angi repeterende grupper av felt (elementer). Dette elementet som et overordnet samleelement og elementet *repeatingGroup* som detaljnivået.

Ingen attributter.

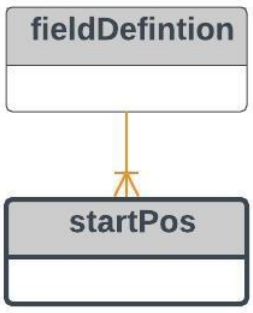
Underliggende elementer	Kravtype	Forekomster
repeatingGroup	mandatory	1 - n

Eksempel:

```
<recordDefinition name=”post1”>
   ….
   <repeatingGroups>
      <repeatingGroup name=”kontigent”>
      ….
      </repeatingGroup>
      <repeatingGroup name=”navn”>
      ….
      </repeatingGroup>
      ….
   </repeatingGroups>
</recordDefinition>
```

Eksemplet viser en posttype “post1” som har to repeterende grupper navngitt som henholdsvis “kontingent» og «navn».

startPos



Elementnavn	Kravtype	Beskrivelse
startPos	optional	Elementet benyttes for å angi startposisjonen for et felt (element).

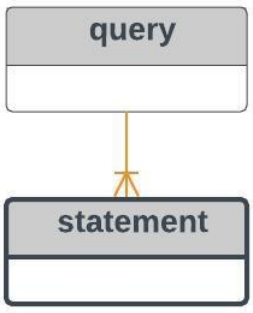
Ingen attributter og ingen underliggende elementer.

Eksempel:

<fieldDefinition name=”**postnr**”>
  <startPos>***1***</startPos>
  <endPos>***4***</endpos>
</fieldDefinition>

Eksemplet viser et element som begynner i posisjon 1 og slutter i posisjon 4.

statement

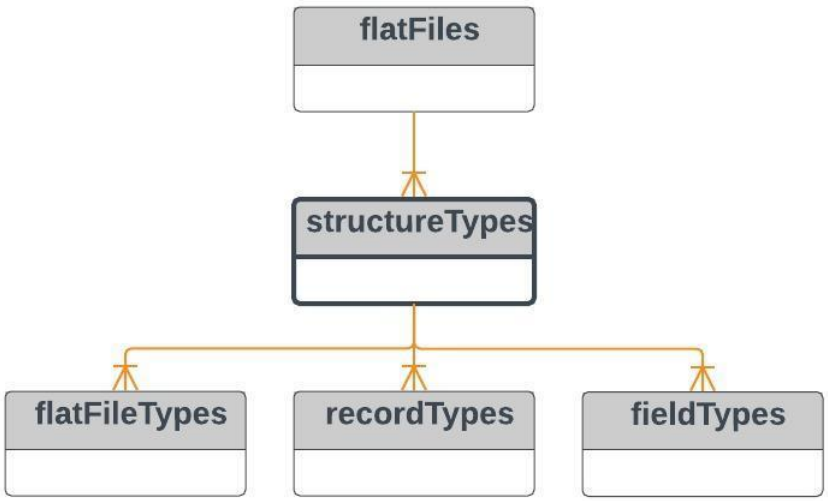


Elementnavn	Kravtype	Beskrivelse
statement	mandatory	I noen henseende kan det være viktig å arkivere hva slags søk som har vært benyttet i den opprinnelige databasen. I dette elementet kan man lagre selve programkodelinjen (statement) som har vært benyttet for et slikt søk. Også andre viktige kodelinjer kan tas vare på i elementet, hvis man ønsker det.

Ingen attributter og ingen underliggende elementer.

For eksempel se under elementet «query».

structureTypes



Elementnavn	Kravtype	Beskrivelse
structureTypes	mandatory	For å minske graden av redundant informasjon i datasettbeskrivelsen og for å forenkle registreringen av den, er det innført typer på de tre hovednivåene – fil, post og felt. Disse typene er samlet under elementet *structureTypes*. En type kan defineres slik at flere fil-, post- eller feltdefinisjoner benytter seg av samme fil-, post- eller felttype respektive. Dermed vil all informasjon på typenivå kun registreres en gang.

Ingen attributter.

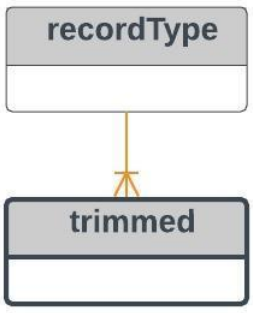
Underliggende elementer	Kravtype	Forekomster
flatFileTypes	mandatory	1
recordTypes	optional	0 - 1
fieldTypes	mandatory	1

Eksempel:

```
<flatFiles>
   ….
   <structureTypes>
      <flatFileTypes>
         <flatFileType name=”typefildef1”>
            ….
         </flatFileType>
      </flatFileTypes>
      <recordTypes>
         <recordType name=”typepostdef1”/>
      </recordTypes>
      <fieldTypes>
         <fieldType name=”typefeltdef1”>
            ….
         </fieldType>
      </fieldTypes>
   </structureTypes>
</flatFiles>
```

I eksemplet er vist hvordan *structureTypes* er benyttet som en beholder for de forskjellige spesifikke type-elementene.

trimmed

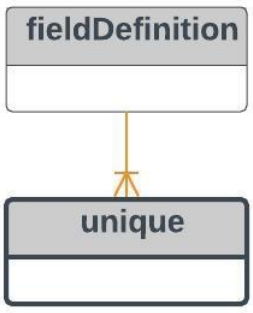


Elementnavn	Kravtype	Beskrivelse
trimmed	optional	Dette elementet benyttes for å angi om en post er trimmet eller ikke.

Ingen attributter og ingen underliggende elementer.

For eksempel se under elementet *recordType*.

unique



Elementnavn	Kravtype	Beskrivelse
unique	optional	Dette elementet benyttes for å angi om verdien i feltet (elementet) er unikt.

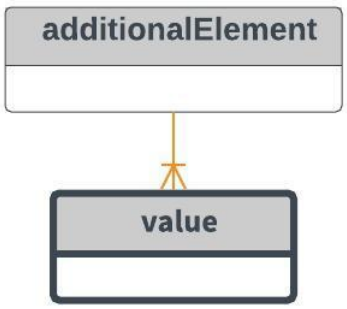
Ingen attributter og ingen underliggende elementer.

Eksempel:

<fieldDefinition name="**fødselsnr**" typeReference="**fnr**">
  <startPos>***1***</startPos>
  <fixedLength>***11***</ fixedLength >
  <notNull/>
  <unique/>
</fieldDefinition>

I eksemplet er vist hvordan man kan angi at et dataelement skal være unikt. I dette tilfelle for et fødselsnummer.

value



Elementnavn	Kravtype	Beskrivelse
value	optional	Dette elementet benyttes i forbindelse med et tilleggsfelt (-element) for å oppgi verdien i feltet (elementet).

Ingen attributter og ingen underelementer.

Eksempel:

<additionalElements>
  <additionalElement name=**recordCreator***>*
    *<value>****Riksarkivet****</value>*
  </additionalElement>
</additionalElements>

I eksemplet er vist bruken av elementet *value* for å tilordne en verdi til et tilleggselement.

Et eksempel på en ADDML fil

```
<?xml version=«1.0» encoding=«UTF-8» standalone=«no»?>
<addml xmlns=«http://www.arkivverket.no/standarder/addml» name =«addmlnavn»>
  <dataset name=«datasettnavn»>
    <reference name=«refnavn»>
      <context>
        <additionalElements>
          <additionalElement name =«recordCreators»>
            <additionalElements>
              <additionalElement name=«recordCreator»>
                <value>Riksarkivet</value>
              </additionalElement>
            </additionalElements>
          </additionalElement>
        </additionalElements>
      </context>
      <content>
        <additionalElements>
          <additionalElement name =«archivalPeriod»>
            <additionalElements>
              <additionalElement name=«startDate»>
                <value>20040401</value>
              </additionalElement>
              <additionalElement name=«endDate»>
                <value>20100331</value>
              </additionalElement>
              <additionalElement name=«period»>
                <additionalElements >
                  <additionalElement name=«inngåendeSkille»>
                    <value>Skarpt</value>
                  </additionalElement>
                  <additionalElement name=«utgåendeSkille»>
                    <value>Mykt</value>
                  </additionalElement>
                </additionalElements>
              </additionalElement>
            </additionalElements>
          </additionalElement>
        </additionalElements>
      </content>
    </reference>
    <flatFiles>
      <flatFile name=«filnavn» definitionReference=«fildef1»>
      </flatFile>
      <flatFileDefinitions>
        <flatFileDefinition name=«fildef1» typeReference=«typefildef1»>
          <recordDefinitions>
            <recordDefinition name=«postdef1» typeReference=«typepostdef1»>
              <keys>
```

```
            <key name=«primnokkel»>
              <primaryKey/>
              <fieldDefinitionReferences>
                <fieldDefinitionReference name=«fodselnr»/>
              </fieldDefinitionReferences>
            </key>
          </keys>
          <fieldDefinitions>
            <fieldDefinition name=«fodselnr» typeReference=«typefeltdef1»>
              <startPos>1</startPos>
              <endPos>11</endPos>
              <unique/>
            </fieldDefinition>
            <fieldDefinition name=«navn» typeReference=«typefeltdef1»>
              <startPos>12</startPos>
              <endPos>41</endPos>
              <notNull/>
            </fieldDefinition>
            <fieldDefinition name=«yrke» typeReference=«typefeltdef1»>
              <startPos>42</startPos>
              <endPos>61</endPos>
              <codes>
                <code codeValue=«statsansatt»/>
                <code codeValue=«kommuneansatt»/>
                <code codeValue=«privat ansatt»/>
                <code codeValue=«arbeidsledig»/>
                <code codeValue=«pensjonist»/>
              </codes>
            </fieldDefinition>
          </fieldDefinitions>
        </recordDefinition>
      </recordDefinitions>
    </flatFileDefinition>
  </flatFileDefinitions>
  <structureTypes>
    <flatFileTypes>
      <flatFileType name=«typefildef1»>
        <charset>utf-8</charset>
        <fixedFileFormat/>
      </flatFileType>
    </flatFileTypes>
    <recordTypes>
      <recordType name=«typepostdef1»/>
    </recordTypes>
    <fieldTypes>
      <fieldType name=«typefeltdef1»>
        <dataType>string</dataType>
      </fieldType>
    </fieldTypes>
  </structureTypes>
```

```
      <flatFileProcesses flatFileReference=«fildef1»>
        <recordProcesses definitionReference=«postdef1»>
          <fieldProcesses definitionReference=«yrke»>
            <processes>
              <process name=«Control_Codes»/>
            </processes>
          </fieldProcesses>
        </recordProcesses>
      </flatFileProcesses>
    </flatFiles>
    <dataObjects>
      <dataObject name=«Rapport»>
        <dataObjects>
          <dataObject name=«rapportfil»>
            <properties>
              <property name=«filnavn»>
                <value>rapport.xml</value>
              </property>
              <property name=«checksum»>
                <properties>
                  <property name=«algoritme»>
                    <value>SHA-256</value>
                  </property>
                  <property name=«verdi»>
                    <value>
F13CED809E4AD36198352495397FABB54DCECCBD5A33BEEDB50BBDD5C9A09232
                    </value>
                  </property>
                </properties>
              </property>
            </properties>
          </dataObject>
          <dataObject name=«skjema»>
            <properties>
              <property name=«filnavn»>
                <value>rapport.xsd</value>
              </property>
              <property name=«checksum»>
                <properties>
                  <property name=«algoritme»>
                    <value>SHA-256</value>
                  </property>
                  <property name=«verdi»>
                    <value>
F13CED809E4AD36198352495397FABB54DCECCBD5A33BEEDB50BBDD5C9A09232
                    </value>
                  </property>
                </properties>
              </property>
            </properties>
```

```
            </dataObject>
          </dataObjects>
        </dataObject>
      </dataObjects>
    </dataset>
</addml>
```