Myndigheten för samhällsskydd och beredskap | RIB

TEKNISK DOKUMENTATIOIN

API för nedkopplad användning av MSB RIB Farliga ämnen

Hur databasen synkroniseras till klienter

Detta dokument är en uppföljning av dokumentet *Dokumentation MSB RIB Farliga ämnen API – Application Programming Interface för webbversionen av MSB RIB Farliga ämnen* (publikationsnummer MSB715 - juli 2018, dnr 2018-06583). Detta dokument ersätter inte det nämnda dokumentet, men det beskriver ett nytt modernare gränssnitt. Båda gränssnitten fungerar för närvarande parallellt.

API för nedkopplad användning av MSB RIB Farliga ämnen
– Hur databasen synkroniseras till klienter

Enheten för arbete med naturolyckor och beslutsstödssystem

Omslagsbild: Magnus Levein
Text: Magnus Levein

Publ nr: MSB1649 - december 2020
ISBN: 978-91-7927-087-2

Innehåll

1 Inledning

Detta dokument vänder sig till utvecklare som vill hämta data från vår databas *MSB RIB Farliga ämnen* på maskinell väg, på samma sätt som vår mobilapp Farliga ämnen hämtar data från MSB:s server.

Lösningen bygger på att mottagaren (klienten) skapar sig en lokal databas, och kommunikationen med MSB:s server syftar till att hålla den lokala databasen uppdaterad.

Kommunikationsprotokollet som används för att kommunicera med MSB:s server är inte lämpligt för att direkt hämta information om en enskild ämnespost. API:ets[1] syfte är i stället att möjliggöra effektiv synkronisering för nedkopplad åtkomst (”off line”) till vår databas.

1.1 Bakomliggande filosofi

I vår administrativa förvaltning av databasen Farliga ämnen använder vi en klassisk SQL-databas med ganska komplicerad struktur. Strukturen är byggd för att göra administrationen effektiv, men den är samtidigt svår att hämta data ur på ett smidigt sätt.

Som exempel kan vi ta ett ämnes kokpunkt, som lagras som två flyttal kompletterat med fyra booleska värden: de två numeriska värdena används för att kunna ange ett kokpunktsintervall; två av de booleska värdena används för att uttrycka ”större än” eller ”mindre än”; de två återstående booleska värdena används för att ange om ämnet sublimerar eller sönderdelas vid den angivna temperaturen.

I stället för att varje program som hämtar data ur Farliga ämnen-databasen själv ska behöva göra alla de logiska kontroller som blir nödvändiga för att kunna presentera de olika värdena korrekt, har vi valt att använda ett mer ”färdigtuggat” överföringsformat i form av en Json-struktur. Det gör vi för vår egen skull. Vi hanterar ju själva flera olika programvaror som läser ur databasen: den installerade Windows-versionen, serverprogramvaran för webbsidan samt mobilappen Farliga ämnen. Under tidigare år har vi råkat ut för buggar som berodde på att en av de olika ”läsarna” läste fel i något sammanhang. Nu vill vi eliminera den möjligheten genom att använda ett centralt ”läsprogram” som hämtar uppgifterna från administrationsdatabasen och distribuerar dem vidare på ett mer lättillgängligt sätt.

[1] API = Application Progamming Interface

En annan viktig aspekt av ”databasen” Farliga ämnen, är att mycket av informationen faktiskt inte administreras i någon databas, utan som fristående filer. Det rör sig framför allt om de textintensiva delarna *Information för räddningstjänsten* och *Akut omhändertagande på olycksplats*, som administreras i form av textfiler och bearbetas till ett slags HTML-liknande filer. Andra exempel på fristående data är bildfilerna som illustrerar olika varningsetiketter, samt förstås hela systemet med hjälp-texter.

För att kunna hantera hela vidden av olika typer av data på ett enhetligt sätt, från Json-strukturen som beskriver en ämnespost till bildfiler för varningsetiketter, har vi valt att bygga den distribuerade databasen som en generell dokumentdatabas. Den valda lösningen innebär att vi skulle kunna göra ganska stora förändringar i ämnesposternas uppbyggnad, utan att för den skull behöva ändra i klient-programvaran.

1.2 Översikt över API:ets olika nivåer

Beskrivningen av API:et är indelad i tre nivåer:

1. **Synkroniseringsnivån** – På denna nivå hanteras all information som tidsstämplade *dokumentposter* som synkroniseras i en generell dokumentdatabas med enkel struktur. Kommunikationen med MSB:s server sker på denna nivå.
2. **Dokumentnivån** – Här förklaras vilka olika dokumenttyper som hanteras i databasen. De enklare dokumenttyperna beskrivs fullständigt här, medan de mer komplicerade dokumenttyperna enbart beskrivs översiktligt.
3. **Detaljnivån** – För de dokumenttyper som kräver en mer detaljerad förklaring ges ett separat kapitel för varje dokumenttyp. Sålunda beskriver ett kapitel ämnespostens Json-struktur, och ett annat kapitel beskriver hur mall-dokumenten är konstruerade.

Rekommenderad läsordning är att läsa kapitlen i den ordning de kommer i dokumentet. Letar man efter specifika detaljer kan man slå upp dem i respektive kapitel, men det förutsätter i princip att man redan har läst hela dokumentet tidigare.

1.3 Eventuella framtida förändringar

Vi bedömer i nuläget att detta API kommer att vara stabilt under många år, i synnerhet vad gäller synkroniseringsnivån. På detaljnivån i Json-strukturen för ämnesposterna kan vi däremot vänta oss att ytterligare värden tillförs, t.ex. för att länka samman ämnesposter med dokument i MSB RIB Bibliotek. Denna typ av förändringar bör dock inte leda till problem med bakåtkompatibiliteten.

Om vi skulle behöva göra så stora förändringar i API:et att bakåtkompabiliteten bryts, är tanken att vi ska publicera det nya API:et under en ny adress och låta både gammalt och nytt API leva parallellt under en övergångsperiod. Detta ska då annonseras genom metadatadokumentet ”version” i databasen, se kapitel 3.2.

2 Synkronisering med MSB:s server

Synkroniseringsnivån – Kapitlet beskriver hur mottagaren (klienten) skapar sig en lokal databas för MSB RIB Farliga ämnen, och sedan håller den uppdaterad i förhållande till MSB:s centrala databas.

2.1 Den synkroniserade databasens struktur

All data hanteras som nyckel-värde-par, och hela datamängden hanteras i en enda SQL-tabell. SQL-tabellens struktur för att hantera vår nyckel-värde-data ser likadan ut på serversidan och på klientsidan:

kolumn ”key”	**kolumn ”deleted”**	**kolumn ”value”**	**kolumn ”timestamp”**
Strängvärde (max 40 tecken) som fungerar som primärnyckel i tabellen.	Bitvärde, där värdet 1 anger att raden är ”raderad”. (För synkroniseringens skull måste alla ”key” som någonsin funnits finnas kvar.)	Lång text, i praktiken Json-formaterad text. (Om raden är ”raderad” är detta fält tomt.)	Tidsstämpel som anger när senaste uppdateringen av denna rad gjordes. 64 bit signed integer, samma värde som DateTime.Ticks enligt UTC.
Denna kolumn ska vara indexerad.			Denna kolumn ska vara indexerad.
Ingen av kolumnerna kan anta NULL-värden.			

Varje rad i datatabellen motsvarar en *dokumentpost*. Som vi ska se i nästa kapitel finns några olika typer av dokumentposter, men ur ett synkroniseringsperspektiv hanteras de alla på samma sätt.

Vid uppdatering på server-sidan kommer endast de dokumentposter som får nya värden att uppdateras, och deras ”timestamp” sätts då till aktuell tidpunkt. Alla poster som uppdateras i en och samma transaktion på serversidan får samma ”timestamp”-värde, i syfte att underlätta eventuell felsökning.

Ett ”key”-värde som någon gång lagts till i tabellen kan inte tas bort, eftersom det skulle förvilla synkroniseringen. I stället kommer flaggan ”deleted” att sättas till 1 och ”value” att tömmas (till tom sträng). Naturligtvis uppdateras också ”timestamp” till aktuell tidpunkt.

Detta upplägg möjliggör inkrementell synkronisering av ändringar i databasen.

2.2 Arbetsflöde när databasen uppdateras på en klient

1. Klienten frågar servern ”Hur många nya dokumentposter finns det, givet att min nyaste dokumentpost har tidsstämpel X? – Och vilket är förresten din nyaste tidsstämpel?” (Om klientens databas är tom skickar den 0 som tidsstämpel.)
2. Servern svarar med antal nya dokumentposter, samt tidsstämpel för den nyaste.
3. Klienten avgör (eventuellt genom att fråga användaren) om den vill hämta de nya posterna eller avvakta. Vilken logik som används här kan bero på sammanhanget.
4. Klienten ber servern ”Skicka mig alla nya dokumentposter, givet att min nyaste dokumentpost har tidsstämpel X”.
5. Servern skickar en Json-array med dokumentposterna. (Uttaget från servern görs med en enda SELECT-sats, isolerad så att den inte påverkas av en eventuell samtidig uppdaterings-transaktion på servern.)
6. Klienten kontrollerar att alla dokumentposter kommit fram.[2] Om de gjort det uppdaterar klienten den lokala databasen med de uppdaterade eller nytillkomna dokumentposterna. Uppdateringen görs som en transaktion.

2.3 Två anrop till MSB:s server

Farliga ämnen-appen använder två olika typer av anrop till MSB:s server för databassynkronisering: **LastUpdateCount** och **GetUpdatedAfter**. Anropen görs som vanliga http GET-anrop, och basadressen till tjänsterna är URL **https://ribdata.msb.se/hazmat/**. Svaren returneras i Json-format.

Vid felaktiga anrop returneras en felsignalerande http-status (typiskt *400 Bad Request* eller *404 Not Found*) samt en Json-kodad felbeskrivning. Värt att notera är att om en obligatorisk parameter utelämnas returneras 404 Not Found, trots att det kanske vore mer beskrivande med en annan statuskod.

De tidsstämplar som används som parametrar är samma heltal (64 bit signed integer) som används i databaskolumnen ”timestamp”. Det är det ”ticks”-värde som C#-funktionen *DateTime.UtcNow* ger. (Se Microsofts .Net-dokumentation för närmare detaljer.)

[2] Minsta möjliga kontroll är att verifiera att Json-strukturen är korrekt, och då speciellt att ”]” som avslutar Json-arrayen finns med. En mer sofistikerad kontroll räknar att antalet dokumentposter stämmer med svaret från föregående anrop. Dock behöver man i så fall ta hänsyn till att servern skulle kunna ha fått fler nya dokument i tidsglappet mellan första och andra anropet.

2.3.1 LastUpdateCount?LastUpdated=*nnn*

Funktionen *LastUpdateCount* anropas med en parameter som anger tidsstämpeln för klientens nyaste dokumentpost. Om klientens databas är tom anger man 0.

Servern svarar dels med hur många uppdaterade poster som finns, och dels vilken tidsstämpel som den nyaste dokumentposten har. Svaret ges som en Json-sträng.

Exempel på fråga

```
https://ribdata.msb.se/hazmat/LastUpdateCount?LastUpdated=637393201679778642
```

Exempel på svar

```
{"LastUpdated":637393967244711940,"Items":1}
```

2.3.2 GetUpdatedAfter?LastUpdated=*nnn*

Funktionen *GetUpdatedAfter* anropas med en parameter som anger tidsstämpeln för klientens nyaste dokumentpost. Om klientens databas är tom anger man 0.

Servern returnerar alla dokumentposter som är nyare än de dokumentposter som klienten redan har. Svaret ges som en Json-array. (Eftersom dokumentposternas data också är Json-kodad blir det en del **\"** i svaret.)

Exempel på fråga

```
https://ribdata.msb.se/hazmat/GetUpdatedAfter?LastUpdated=637393201679778642
```

Exempel på svar

```
[{"key":"version","value":"{\"UseUntil\":\"2021-06-30\",\"UpdateApp\":\"\"}",
"timestamp":"637393967244711940","deleted":false}]
```

Notera att kolumnen ”timestamp” i exemplet överförs som en Json-sträng och inte som ett tal, och att ”deleted” anges som true/false och inte som en siffra. Det är dock klokt att bygga en implementation så att den fungerar oavsett om ”timestamp” överförs som sträng eller som tal, och även om ”deleted” skulle råka komma som en siffra (1 eller 0).

Teckenkodningen som används är UTF-8.

2.4 Möjlighet till kontroll och felsökning

Ett enkelt sätt att kontrollera om databasen synkroniserats korrekt är att göra en sammanställning över vilka tidsstämplar som finns, och hur många dokumentposter som har respektive tidsstämpel.

Uttryckt med SQL kan det till exempel se ut på detta sätt:

```
SELECT timestamp, count(*) FROM Hazmat GROUP BY timestamp ORDER BY 1 DESC;
```

I vår app Farliga ämnen skrivs just denna information ut för de tio senaste tidsstämplarna, om man väljer Övrigt > Databas > Detaljer för felsökning. (Tidsstämplarna skrivs där ut som datum och klockslag i UTC.)

3 Databasens dokumenttyper

Dokumentnivån – Kapitlet beskriver de olika dokumenttyper som förekommer i den distribuerade Farliga ämnen-databasen.

3.1 Dokumenttypen framgår av nyckelnamnet

Som vi såg i föregående kapitel är den synkroniserade databasen uppbyggd som en nyckel-värde-lagring, där varje rad i databasen har en nyckel (kolumnen ”key”) och ett värde (kolumnen ”value”).

För att på ett enkelt sätt särskilja olika typer av dokumentposter inleds nyckelnamnen med prefix, på ett sätt som påminner om en katalogstruktur. (Det speciella metadatadokumentet ”version” har inget egentligt prefix, eftersom det alltid bara förekommer ett sådant dokument.) Följande prefix används:

Prefix	**Fullständigt nyckelnamn (exempel)**	**Förklaring**
version	version	En singulär post med versionsinformation för hela databasen. Json.
css/	css/app.css	Stilmallar (CSS) sparade som text inbäddad i Json.
help/	help/fahjalp.html	Hjälpfiler sparade som HTML inbäddad i Json.
img/	img/etk_2_3.png	Bildfiler (PNG) med tillhörande förklaringstext, sparade i en Json-struktur med Base64-kodad binärdata.
doc/	doc/448	Ämnesposter sparade i Json-format.
tpl/	tpl/sirib45.htm	Malldokument sparade som HTML i *Liquid Templating Language*, inbäddade i Json.

De olika dokumenttyperna förklaras närmare i de följande underkapitlen.

3.2 Metadatadokumentet ”version”

Databasen innehåller ett särskilt metadatadokument i den dokumentpost som har nyckelnamnet ”version”. Syftet med detta metadatadokument är att vi ska kunna hantera två särskilda situationer:

a) Användaren har en databas som aldrig blir uppdaterad, t.ex. för att enheten inte är ansluten till internet. Då ska ett varningsmeddelande visas när databasen passerat sitt ”bäst före-datum”.

b) Vi vill även kunna varna användare för att de måste uppdatera sin programvara. Det kan till exempel vara så att vi bygger om datastrukturen så att kompatibiliteten bryts. Även då ska ett varningsmeddelande visas när användaren har kontakt med vår server.

Metadatadokumentet är en Json-struktur med två värden, *UseUntil* och *UpdateApp*. Innehållet ser typiskt ut på detta sätt:

```
{"UseUntil":"2021-06-30","UpdateApp":""}
```

Värdet **UseUntil** är databasens bäst före-datum. Klienten förväntas jämföra detta med dagens datum enligt lokal tid. Om bäst före-datumet har passerats finns det risk att allt som står inte längre gäller. Klienten ska då påtala för användaren att **databasen är för gammal och behöver uppdateras** eftersom regelverk kan ha ändrats. Användaren ska dock tillåtas att fortsätta använda sin databas på egen risk.

Värdet **UpdateApp** är i normalfallet tomt. Om det innehåller någon text förväntas klienten visa denna text för användaren som ett varningsmeddelande. Om vi behöver göra så stora förändringar i API:et att bakåtkompabiliteten bryts lägger vi ett meddelande i detta fält innan vi upphör med uppdateringarna. Användarna kan på det sättet uppmanas att uppdatera sin programvara. Den nya versionen av programvaran läser från en annan URL och därmed ur en annan datatabell. Jämför kapitel 1.3.

3.3 Stilmallar som ”css/”

Dokumenttypen ”css” används för de stilmallar i CSS som MSB RIB Farliga ämnen använder. CSS är till sin natur en textfil, men i databasen är texten inbäddad i Json på detta sätt:

```
{"data":"/* CSS text */"}
```

För närvarande finns endast dokumentet ”css/app.css”, men det kan inte uteslutas att det så småningom kompletteras med fler CSS-filer, t.ex. för andra plattformar.

3.4 Hjälpfiler som ”help/”

Dokumenttypen ”help” avser filer som används av hjälpsystemet som är integrerat i MSB RIB Farliga ämnen.

Hur hjälpsystemet fungerar ingår inte i API-definitionen och kan ändras utan förvarning. I bilaga A ges dock några detaljer om den nuvarande konstruktionen.

3.5 Bilder + alternativtext som ”img/”

Bilder lagras som dokumenttyp ”img”. Eftersom varje bild behöver ha en alternativtext associerad till sig används en enkel Json-struktur som inkluderar både alternativtext och bilddata:

```
{"title":"Giftiga gaser","data":"iVBORw0KGgoAAAANSUhEUgAAAFQAAABUCAYAAAFrbCB
XAAAACXBIWXMAAAsTAAALEwEAmpwYAAAABGdBTUEAALGOfPtRkwAAACBjSFJNAAB6JQAAgIMAAPn
/.../
IgMfxMwAkqtdfMqw9kQAAAABJRU5ErkJggg==}
```

Exemplet visar dokumentposten ”img/etk_2_3.png”. Bildfilen är en PNG-fil kodad som Base64. (För närvarande är alla bildfiler PNG-filer, vilket återspeglas i att dokumentposternas nyckelnamn slutar på ”.png”, men även andra filformat är förstås tänkbara i framtiden.)

3.6 Ämnesposter som ”doc/”

Kärnan i Farliga ämnen-databasen är förstås själva ämnesposterna. De lagras som dokumenttyp ”doc”. Dokumentposternas fullständiga nyckelnamn utgörs av prefixet ”doc/” följt av ämnespostens unika ämnes-id.

Ämnes-id är primärnyckel i vår administrationsdatabas, och är alltså ett internt nummer som unikt identifierar en ämnespost; just nu 1~10 000 men det kan förstås växa uppåt i rimliga proportioner.

Ämnesposten lagras som en ganska lång Json-struktur, som innehåller flera olika typer av data. En detaljerad beskrivning ges i kapitel 4. Ett komplett exempel ges i bilaga B.

3.7 Malldokument som ”tpl/”

Varje innehållsflik (som t.ex. Akutvård, Fysdata eller Miljö) som visas i Farliga ämnen-appen är byggd på en mall. Vissa flikar använder en och samma statiska mall oavsett ämnespost, medan andra flikar använder olika mallar för olika ämnesposter.

Speciellt kan nämnas att flikarna Räddning och Akutvård innehåller faktatexter i mallarna, anpassade på så sätt att olika mallar används för olika grupper av ämnen. (Detta sätt att använda malldokumenten på skulle mycket väl kunna byggas ut till att omfatta fler flikar i framtiden, t.ex. fliken Miljö.)

Varje ämnespost inkluderar information om vilka malldokument som ska användas för just den ämnesposten. Klientprogramvaran behöver därför inte ha någon närmare kunskap om ifall det är fråga om en statisk mall eller en mall som bara används av en specifik ämnesgrupp.

Malldokumenten är HTML-filer berikade med *Liquid Templating Language*, vilket innebär att de innehåller platshållare (referenser) till ämnespostens olika data-värden. De innehåller också viss logik med villkor för att t.ex. dölja eller visa vissa

rubriker eller statiska texter beroende på den aktuella ämnespostens olika data-värden.

Ämnesposterna (”doc/”) och malldokumenten (”tpl/”) fungerar alltså i symbios med varandra: ämnesposterna hänvisar till vilka malldokument som ska användas för det aktuella ämnet, och malldokumenten ger närmare instruktioner om vilka av ämnespostens datavärden som ska presenteras var, när och hur.

Denna konstruktion gör systemet mycket flexibelt, och det är möjligt att t.ex. införa helt nya datavärden i ämnesposten och få dem korrekt presenterade för användarna utan att behöva göra några ändringar i klientprogramvaran.

Malldokumenten är alltså textfiler till sin natur, men i databasen är texten inbäddad i Json på detta sätt:

```
{"data":"<!-- HTML-dokument i Liquid Templating Language -->"}
```

I kapitel 5 ges närmare detaljer om malldokumenten.

3.8 Typsnittsfiler inkluderas inte

Som framgår av ovanstående genomgång inkluderas ganska många typer av data i den distribuerade Farliga ämnen-databasen. Däremot inkluderas inte de typsnitts-filer som CSS-stilmallarna refererar till.

Exakt vilka typsnitt som de inkluderade CSS-mallarna hänvisar till, och som behövs för att stilsättningen ska fungera som avsett, ingår inte i API-definitionen utan kan ändras från tid till annan.

Dock kan nämnas att det i skrivande stund är typsnitten *FontAwesome5FreeSolid*[3], *Open Sans*[4] och *Poppins*[5] som utnyttjas, varav FontAwesome5FreeSolid är speciellt viktig eftersom den används för pilar och andra typer av symboler. (Open Sans och Poppins kan ersättas av andra typsnitt utan problem.)

[3] Förväntad filsökväg: **fonts/fa-solid-900.woff**
[4] Förväntade filsökvägar: **fonts/Open_Sans_400.woff**, **fonts/Open_Sans_400i.woff**, **fonts/Open_Sans_600.woff**, **fonts/Open_Sans_600i.woff**
[5] Förväntad filsökväg: **fonts/Poppins_600.woff**, och för hjälpfilen dessutom **fonts/Poppins_400.woff**

4 Ämnesposternas Json-struktur

Detaljnivån – Kapitlet beskriver ämnesposternas Json-struktur, och de olika datatyper som den består av. (Hur denna Json-struktur lagras i dokumentdatabasen förklarades i kapitel 3.6.)

4.1 Översiktlig struktur

Nedanstående strukturbeskrivning avser hur datamängden för en (1) ämnespost uttrycks i Json, inklusive hänvisningarna till sektionsspecifika HTML-mallar.

När ämnesposterna visas för användaren är de tematiskt indelade i sektioner (”flikar”) vilket dock inte påverkar ämnespostens datastruktur, mer än att varje sektion behöver en hänvisning till sin sektionsspecifika HTML-mall.

Datastrukturen är i huvudsak generisk. De generiska variablerna har en av tre tillgängliga datatyper: *värde*, *array* eller *nyckelförsedd array*. För att hantera dels **synonymer** och dels **hygieniska gränsvärden** används dock specifika strukturer. Tanken med att använda så mycket generiska variabler som möjligt är att vi i ett senare skede ska kunna lägga till en ny variabel utan att behöva göra någon ändring i applikationen (så länge någon av de tre generiska datatyperna används).

Ämnespostens grova struktur är följande:

```
{
    "ID": …,
    "Templates": { … },
    "data" : { … }
}
```

Under ”data” döljer sig både generiska ämnesdata och de två specifika strukturerna för synonymer och för hygieniska gränsvärden.

I bilaga B redovisas ett verkligt exempel på en komplett ämnespost: Ammoniak, vattenfri (UN 1005).

4.2 Unik nyckel (”ID”)

Ämnespostens ämnes-id är utbruten ur den generiska strukturen, eftersom den utgör ämnestabellens unika nyckel. Ämnes-id är ett heltal, just nu 1~10 000 men det kan förstås växa uppåt i rimliga proportioner. Jfr kapitel 3.6

4.3 Referenser till sektionsmallarna (”Templates”)

Ämnespostens referenser till HTML-mallarna är också utbruten ur den generiska datastrukturen. Referensobjektet ser ut på detta sätt:

```
"Templates": {
     "Identitet":"…",
     "Raddning":"…",
     "Akutvard":"…",
     "Fysdata":"…",
     "Miljo":"…"
}
```

Värdena som anges är mallarnas namn utan prefix. För Akutvård kan värdet t.ex. vara `"sirib45.htm"`, vilket betyder att den ifrågavarande mallen lagras i dokument-databasen under nyckeln ”tpl/sirib45.htm”.

På sikt är det mycket möjligt att fler sektioner (”flikar”) tillförs enligt samma mönster.

4.4 Ämnespostens egentliga data (”data”)

4.4.1 Data med datatypen *värde*

Datatypen *värde* omfattar data som uttrycks med ett enstaka värde. Värdet kan vara ett heltal, ett flyttal eller en sträng. De flesta fält kan också vara tomma, vilket uttrycks som en tom sträng. (Några av fälten kan uteslutas helt, vilket anges särskilt nedan.)

Bokstäverna som förekommer i kolumnen Anmärkning förklaras direkt efter tabellen.

Parameternamn	Anm.	Beskrivning
Namn		Ämnespostens huvudnamn. Fältet har alltid ett värde.[6]
Fortydligande		Ämnespostens förtydligandetext. Förtydligar vad som menas med ämnespostens namn. Huvudsyftet är att hjälpa användaren att välja rätt ämnespost.[7]
LosningAvId		Om ämnesposten är en lösning av en annan ämnespost anges det lösta ämnets id här. I annat fall är fältet tomt.
Beskrivning		Koncentrerad beskrivning av ämnets egenskaper grundat på dess transportklassificering.[8]
FarlNr		Farlighetsnummer enligt ADR eller RID. Kan utöver siffror innehålla bokstav och punkt.
FarlNrTxt		Textförklaring av vad farlighetsnumret betyder. (Fältet kan i teorin vara tomt även om FarlNr är angivet.)
FarlNrADR		Farlighetsnummer enligt ADR. Kan utöver siffror innehålla bokstaven X. (Detta värde är för flera ämnesposter tomt även om *FarlNr* är angivet.)

[6] (Intern kommentar: detta värde heter *amne_ben* i vår administrationsdatabas.)
[7] (Intern kommentar: detta värde heter *amne_beskr* i vår administrationsdatabas.)
[8] (Intern kommentar: detta värde heter *rtjinfo_beskr* i vår administrationsdatabas.)

FgADR		Anger för respektive regelverk (ADR, RID, IMDG, IATA-DGR) om ämnesposten omfattas av reglerna: 1: Ej bedömt 2: Omfattas ej av regelverket 3: Farligt gods 4: Transport ej tillåten
FgRID		
FgIMDG		
FgIATADGR		
Klass		Klass enligt ADR.[9]
Klassif		Klassificeringskod enligt ADR.[10]
Fpg		Förpackningsgrupp enligt ADR.[11]
UN	A	Om ämnesposten motsvarar en rad i transport-regelverket anges UN-numret här, t.ex. ”0034”. I annat fall är fältet utelämnat.
UNutg	A	Om ämnesposten motsvarar ett UN-nummer som tidigare fanns i transportregelverket, men som tagits bort, anges numret här, t.ex. ”3050”. I annat fall är fältet utelämnat.
Anm		Anmärkning om ämnesposten.[12] Ibland är det ett förtydligande till någon uppgift, ibland är det mer allmän information.
AggTill		Aggregationstillstånd.
Farg		Färg.
Lukt		Lukt.
SmaltP	B	Smältpunkt.
KokP	B	Kokpunkt.
BrbOmr		Brännbarhetsområde, t.ex. ”från 15 till 28 vol-%” eller ”Oxiderande (brandunderstödjande)”.
FlamP	B	Flampunkt.
TermTP	B	Termisk tändpunkt.
Dens		Densitet. Avser flytande eller fast form av ämnet.
DensTal		Densitetstal, det vill säga hur tunga gaser eller ångor från ämnet är i förhållande till vanlig luft, som har densitetstal 1,0.
Visk		Viskositet (kinematisk viskositet).
VtnLosl		Vattenlöslighet. Anges oftast både med ord och med siffror (viktprocent).
Molekyl		Molekylformel i textformat. Klientapplikationen bör snygga till den innan den visas för användaren. Se bilaga C.

[9] Alla ämnesposter har likalydande ADR-klass och RID-klass.
[10] Det finns en ämnespost med skillnad i klassificeringskoden mellan ADR och RID: UN 1043 saknar klassificeringskod i RID. (Enligt kontroll i ADR/RID 2019.)
[11] Det finns två ämnesposter där förpackningsgruppen skiljer sig åt mellan ADR och RID: UN 3533 och 3534 saknar förpackningsgrupp i RID. Det beror på att järnvägstransport av dessa ämnen är helt förbjuden.
[12] (Intern kommentar: detta värde heter *amne_anm* i vår administrationsdatabas.)

MolVikt		Molekylvikt i g/mol.
JonPot		Jonisationspotential.
KritT		Kritisk temperatur i grader Celsius.
AngtryckA		Konstanterna A [N/m²] och B [K] i formeln som beskriver ämnets ångtryckskurva: $P_S(T) = A \cdot e^{\frac{-B}{T}}$ P_S är ångtrycket [Pa] vid given temperatur T [K].
AngtryckB		
AngtryckPkt	B	Ångtrycksangivelse som punktvärde (så som det är inmatat i databasen).
Riskomrade		Rekommenderat initialt riskområde. Om värdet är tomt beror det antingen på att rekommendation saknas, eller på att en riskområdestabell visas från HTML-mallen i stället (jfr kapitel 5.2.2).
Skyddsutrustning		Rekommenderad skyddsutrustning vid läckageplats. (Texten som anger rekommenderad skyddsutrustning vid livräddning styrs från HTML-mallen.)
Slackmedel		Släckmedelsrekommendation.
GhsSignord		Signalord enligt harmoniserad CLP-märkning. Antingen `FARA` eller `VARNING`. Fältet kan vara tomt.
GhsFinns		Om ämnet har information om harmoniserad CLP-märkning anges `X` (som representerar ”sant”). Annars tom sträng.
GhsAnm		Eventuell anmärkningstext rörande informationen om harmoniserad CLP-märkning; CLP-informationen kan bli missvisande om inte anmärkningen visas. Fältet kan vara tomt. Exempelämne: UN 3318.

A) Jämför **UNovr** under datatypen *array*. Endast ett av **UN**, **UNutg** och **UNovr** kan ha värde satt.

B) Vissa fysikaliska uppgifter kan ges med mindre än- eller större än-tecken (<, >). I Json-strukturen är de lagrade just så, även om det innebär att de måste kodas om till `<` respektive `>` om de ska gå att använda i HTML-kod.

4.4.2 Data med datatypen *array*

Datatypen *array* omfattar de data som kan uttryckas med en sträng-array (inklusive specialfallet att strängarna utgör filnamn till bilder). Skillnaden mot enkla värden är att i arrayer kan det vara flera värden för en och samma parameter. Värdena är strängar, men fält kan också utelämnas helt om värde saknas.

Parameternamn	Anm.	Beskrivning
UNovr	A	Om ämnesposten inte motsvarar någon rad i transportregelverket, men ändå måste transporteras under samlingsbenämning, anges vilken eller vilka samlingsbenämningar som kan komma ifråga som UN-nummer här, t.ex. [”2780”] eller [”2779”, ”2780”, ”3013”]. I annat fall är fältet utelämnat.
CAS		Ämnespostens samtliga CAS-nummer[13], t.ex. [”64742-47-8”, ”68334-30-5”, ”68476-30-2”]. Fältet utelämnas om CAS-nummer saknas.
EG		Ämnespostens samtliga EG-nummer, t.ex. [”265-149-8”, ”269-822-7”, ”270-671-4”]. Fältet utelämnas om EG-nummer saknas.
Reaktioner		Ämnesspecifika riskfaktortexter (reaktionstexter). Fältet utelämnas om texter saknas.
AkutvardInfo		Ämnesspecifika akutvårdstexter.[14] Fältet utelämnas om texter saknas.
TEtiketter		Filnamn på transportetiketter för etikettering av kollin enligt ADR och RID. Fältet utelämnas om transportetiketter saknas.
GhsPiktogram		Filnamn på faropiktogram enligt harmoniserad CLP-märkning. Fältet utelämnas om faropiktogram saknas.
Anvomr		Ämnets användningsområden. Fältet utelämnas om texter saknas.

A) Jämför **UN** och **UNutg** under datatypen *values*. Endast ett av **UN**, **UNutg** och **UNovr** kan ha värde satt.

4.4.3 Data med datatypen *nyckelförsedd array*

Datatypen *nyckelförsedd array* är en utökad array. Varje element i arrayen består av ett beskrivning-värde-par. Man kan betrakta det som en enkel tabell med två kolumner: den vänstra kolumnen är en slags beskrivning eller kod, och den högra kolumnen är ett värde eller en uttydning av koden.

I Json representerar vi detta som en array av objekt, där varje tabellrad utgör ett objekt:

```
"ToxTolkning":
[
    { "Akut giftighet": 1 },
    { "Anrikas i naturen": 3 },
    { "Cancerframkallande": 2 },
    { "Långsiktiga skador på organ/nervsystem": 2 }
]
```

[13] (Intern kommentar: detta omfattar både nummertyp *CAS* och *CAS_X* i vår administrationsdatabas.)
[14] (Intern kommentar: egenskaper med kryssrutan ”sjukvård” ikryssad i vårt administrationssystem.)

I beskrivnings-exemplen nedan ges (av utrymmesskäl) bara en tabellrad, även om tabellerna ofta har fler rader än så.

Parameternamn	Anm.	Beskrivning
ToxTolkning		Tabellen ”Tolkning av toxikologiska data” på Miljö-fliken. 1: ”Ja” 2: ”Nej” 3: ”Okänt” `[ {"Akut giftighet": 1 } ]`
ToxData	B	Tabellen ”Toxikologiska data” på Miljö-fliken. `[ {"Giftighet vid inandning: LC50 inhalerat mus, 1 h": "> 0,4 mg/l" } ]`
GhsFaror		Faroangivelser enligt harmoniserad CLP-märkning. `[ {"H400": "Mycket giftigt för vattenlevande organismer." } ]`

B) Vissa fysikaliska uppgifter kan ges med mindre än- eller större än-tecken (<, >). I Json-strukturen är de lagrade just så, även om det innebär att de måste kodas om till `<` respektive `>` om de ska gå att använda i HTML-kod.

4.4.4 Specifik struktur för synonymer

Ämnespostens information om synonymer är tyvärr för komplicerad för att kunna redovisa med en generisk datastruktur. Därför används i stället följande specifika struktur: en array med synonym-objekt.

Den obligatoriska beståndsdelen i synonym-objektet är ett nyckel-värde-par med språkkod som nyckel och själva synonymen som värde. Utöver detta kan även en typ-angivelse ges: 1=ADR, 2=Handelsnamn, 3=ADR och Handelsnamn.

Språkkoderna väljs enligt ISO 639-1, vilket i vårt fall blir dessa fyra: en, de, fr, sv.

Ett enkelt exempel visar hur det skulle kunna se ut:

```
"Synonymer":
[
    {"en": "Ammonia"},
    {"de": "Ammoniak, wasserfrei", "typ": 1},
    {"sv": "R 717", "typ": 2}
]
```

4.4.5 Specifik struktur för hygieniska gränsvärden

Ämnespostens information om hygieniska gränsvärden och lukttrösklar hanteras i ett specialdesignat objekt. Objektet består av upp till 20 nyckel-värde-par. Många av värdena gäller för en särskild exponeringstid, och då anges tiden i minuter som ett suffix i nyckelnamnet (efter ’_’).

Studera gärna hur tabellen som presenterar hygieniska gränsvärden ser ut i MSB RIB Farliga ämnen, men använd en tillräckligt bred skärm för att se den fullständiga tabellen. (På telefoner kommer appen Farliga ämnen att visa en omgjord layout.)

För närmare förklaring om de olika gränsvärdena hänvisas till den inbyggda hjälpen i MSB RIB Farliga ämnen.

Följande nycklar är definierade för de tre raderna ”Risk för …”. Dock är det bara en av grupperna *a–c* som används i varje enskilt fall:

a) AEGL1_10, AEGL1_30, AEGL1_60, AEGL1_240, AEGL1_480, AEGL2_10, AEGL2_30, AEGL2_60, AEGL2_240, AEGL2_480, AEGL3_10, AEGL3_30, AEGL3_60, AEGL3_240, AEGL3_480

b) ERPG1_60, ERPG2_60, ERPG3_60

c) TEEL1_15, TEEL2_15, TEEL3_15

För raden ”Arbetsmiljö” används dessa nycklar; maximalt kombineras en KGV-variant med NGV-värdet:

- KGV(V)_5, KGV(V)_15, KGV_5, KGV_15, NGV_480

Notera skillnaden på KGV = bindande korttidsgränsvärde som inte får överskridas, och KGV(V) = vägledande korttidsgränsvärde som är ett rekommenderat högsta värde.

De återstående raderna ”Filtermask ej lämplig” (= IDLH), ”Uttalad lukt” och ”Förnimbarhet” – som alla saknar tidsangivelse – har dessa nycklar:

- IDLH, Lukt, Fornim

Värdena anges alltid i ppm som siffervärden (inte som strängar). När de presenteras för användaren skrivs däremot alltid enheten ”ppm” ut.

Ett enkelt exempel hämtat från kolmonoxid visar hur det kan se ut:

```
"Gransvarden":
{
    "AEGL3_10": 1700,  "AEGL3_30": 600,  "AEGL3_60": 330,
    "AEGL3_240": 150,  "AEGL3_480": 130,  "AEGL2_10": 420,
    "AEGL2_30": 150,  "AEGL2_60": 83,  "AEGL2_240": 33,
    "AEGL2_480": 27,  "KGV(V)_15": 100,  "NGV_480": 35,
    "IDLH": 1200
}
```

5 Malldokumenten

Detaljnivån – Kapitlet ger fördjupad information om hur malldokumenten i Farliga ämnen-databasen fungerar.

5.1 Varför malldokument?

MSB RIB Farliga ämnen innehåller olika typer av information som inte kan presenteras på ett bra sätt i ren textform:

- Orange farligt gods-skylt och transportmärkning (etiketter) för farligt gods, samt faropiktogram enligt CLP.
- Åtgärdsschema för räddningstjänsten, som kan grena upp sig beroende på om det är brand eller läckage.
- Interaktiv ångtryckskurva med möjlighet att välja ämnets temperatur, och då få temperaturspecifika uträkningar av ångtryck, flyktighet och mättnadskoncentration.
- För vissa ämnen behöver det rekommenderade initiala riskområdet ges som en tabell. (Men det vanligaste är att det består av en enkel text.)

Dessutom finns längre, mer resonerande, texter på vissa av innehållsflikarna,[15] och de texterna behöver kunna ha formatering som t.ex. fet stil eller hyperlänkar till webben.

Eftersom vi vill kunna visa innehållet i Farliga ämnen-databasen på flera olika plattformar, med så lite dubbelarbete som möjligt, har vi valt att använda HTML med CSS och Javascript som presentationsteknik. Utifrån ett utvecklarperspektiv är det en enkel textbaserad lösning, men den är ändå kraftfull nog att hantera de olika informationstyperna vi behöver hantera (enligt ovan). HTML fungerar rakt av på webben, och det kan läsas in i webbvyer i såväl Windows-applikationer som mobilappar.

För att kunna presentera med HTML behöver HTML-koden genereras på något sätt. Enklaste sättet att göra det är genom att definiera någon slags skelett som vår programvara får fylla på med anpassat innehåll för det aktuella tillfället. Malldokumenten utgör just detta skelett. Som vi ska se i nästa avsnitt kan skelettet vara mer eller mindre innehållsrikt.

5.2 Malldokumentens dubbla funktion

Alla malldokument fungerar som en strukturell grund för att definiera vilka uppgifter som ska visas var. Men vissa malldokument innehåller också faktatexter i sig själva.

[15] I nuläget är det flikarna Räddning och Akutvård som inkluderar längre texter på detta sätt.

5.2.1 Funktion som strukturell grund

Utifrån resonemanget i 5.1 ser vi att ett viktigt syfte med malldokumenten är att tjäna som strukturell grund för den slutliga HTML-sida som ska presentera information om ett valt ämne. Malldokumentet ”tpl/fysdata.htm” fungerar på just detta sätt (här återgivet i förkortad form, och innan minifiering):

```
<h1>Fysikaliska data</h1>
  {% if Anm %}
    <div class="anmarkning">
       <h3>Anmärkning:</h3><p>{{ Anm }}</p>
    </div>
  {% endif %}
  <h2>Allmänt</h2>
    <h3>Tillstånd:</h3><p><span data-rib="AggTill">{{ AggTill }}</span></p>
    <h3>Färg:</h3><p><span data-rib="Farg">{{ Farg }}</span></p>
    <h3>Lukt:</h3><p><span data-rib="Lukt">{{ Lukt }}</span></p>
    <div class="tomrad"></div>
    <h3>Smältpunkt:</h3><p><span data-rib="SmaltP">{{ SmaltP }}</span></p>
     <!-- ... -->

  <h2>Förångnings&shy;egenskaper</h2>
    {% if AngtryckA and AngtryckB %}
      <!-- ... -->
    {% endif %}

  <h2>Gränsvärden</h2>
    {% assign AeglErpgTeel = nil %}
    {% assign Col_5 = GRV_KGVV_5 or GRV_KGV_5 %}
    {% assign Col_10 = false %}
    <!-- ... -->

<script>
  // Ångtrycksberäkningar i MSB RIB Farliga ämnen
  /* ... */
  var MSB_RIB_VaporPressure = (function() {
    /* ... */
  })();
</script>
```

Vi kan se att sidan inleds med en H1-rubrik ”Fysikaliska data”. Därefter visas eventuellt en anmärkningstext, och sedan kommer första delsektionen ”Allmänt”. Den börjar med rubriken ”Tillstånd”, och där visas ämnets aggregationstillstånd (som enligt 4.4.1 har parameternamnet *AggTill*). Och så fortsätter det. Sist på sidan finns det javascript som ritar ut ångtryckskurvan och som låter användaren välja ämnets temperatur.

Malldokumenten använder *Liquid Templating Language* för att bädda in logiska villkor och referera till ämnespostens datavärden. Vi återkommer till det i kapitel 5.3.

5.2.2 Funktion som faktaartikel

För flikarna Räddning och Akutvård fungerar malldokumenten inte bara som strukturell grund, utan även som en gemensam faktagrund för en specifik ämnesgrupp. Sålunda finns, med ungefärliga siffor, 50 malldokument för Räddning och 40 malldokument för Akutvård. Samma teknik kan mycket väl komma att användas för fler flikar i framtiden, t.ex. fliken Miljö.

Malldokumenten för Räddning är indelade i kategorier baserat på ämnenas transportklassificering (t.ex. ”Brandfarliga gaser”, ”Giftiga ämnen”, ”Självantändande ämnen”), medan malldokumenten för Akutvård är indelade efter deras förgiftningsegenskaper (t.ex. ”Aromatiska föreningar”, ”Retande gaser”, ”Sexvärda kromföreningar”).

För dessa två flikar utgör faktatexterna i mallarna sidans huvudsakliga innehåll. Mallarnas lite mer övergripande faktatexter kompletteras med ämnesspecifika detaljer (med hjälp Liquid Templating Language, precis som ovan).

Som exempel tittar vi närmare på ett utdrag ur mallen ”tpl/r_f.htm” som avser fliken Räddning för brandfarliga ämnen:

```
<div id="irinf">
  <h2>Initial information</h2>
    <h3>Ämnesbeskrivning:</h3><p>{{ Beskrivning }}</p>
    <h3>Initialt riskområde:</h3><p class="tonad">{{ Riskomrade }}</p>
    <h3>Skydd – livräddning:</h3><p>Branddräkt och tryckluftsapparat.</p>
    <h3>Skydd – läckageplats:</h3><p class="tonad">{{ Skyddsutrustning }}</p>
    <h3>Släckmedel:</h3><p>{{ Slackmedel }}</p>
</div>

<div id="riskfaktorer">
  <h2>Riskfaktorer <span class="nobold">– Brandfarliga ämnen</span></h2>
    <p>Brandrisk. En brand kan sprida sig snabbt. Risk för återantändning
    efter släckt brand.</p>
    <p>För <strong>vätskor</strong> gäller att brännbara ångor kan sprida sig,
    och ge explosionsrisk vid läckage inomhus eller i avlopp. Ångorna är i
    många fall tyngre än luften (kontrollera på Fysdata-fliken!) och kan då
    samlas i lågt belägna utrymmen, t.ex. källare. [...]</p>
    <!-- ... -->

  {% if Reaktioner %}
    <div class="tonad">
      <h3>Särskilt för <em>{{ Namn }}:</em></h3>
        <ul>
          {% for text in Reaktioner %}
            <li>{{ text }}</li>
          {% endfor %}
        </ul>
    </div>
  {% endif %}
```

Lägg särskilt märke till två saker:

- Texten för **Skydd – livräddning** är styrd av mallen, medan texten för **Skydd – läckageplats** hämtas från ämnespostens datastruktur.
- Texten under **Riskfaktorer** börjar med en flera stycken lång text som står direkt i mallen, men kompletteras sedan med en uppräkning av ämnespostens specifika riskfaktortexter (reaktionstexter), om sådana finns.

Just på fliken Räddning påverkar valet av malldokument även om **Initialt riskområde** ska visas som en text från ämnesposten (med parameternamn *Riskomrade*, som i exempelkoden), eller om det ska visas som en tabell – vilken i så fall är kodad direkt i malldokumentet.

5.3 Hur ämnespostens data görs tillgänglig för malldokumenten

Vi använder templating-motorn *Fluid.Core* för att göra om malldokumenten från mallar skrivna i *Liquid Templating Language* till färdig HTML-kod.[16]

5.3.1 Mappning av datatyper

De olika datavärdena tillgängliggörs för templating-motorn under samma namn som de definieras med i Json-strukturen. Detta sker på ett generiskt sätt, så att strukturen kan byggas ut med fler värden utan att någon omprogrammering behöver göras:

- Datatypen *värde* tillgängliggörs som Liquid-datatypen ”string”.
- Datatypen *array* tillgängliggörs som Liquid-datatypen ”array”.
- Datatypen *nyckelförsedd array* tillgängliggörs som Liquid-datatypen ”array av array”, där nyckeldelen av varje rad har index [0] och värdedelen av varje rad har index [1].

För att förenkla testvillkoren i Liquid-koden tilldelas dock inga tomma värden: om Json-strukturen innehåller ett tomt värde, så kommer motsvarande variabel i Liquid *inte* att tilldelas.[17] Detta medför att villkoret `{% if variabelnamn %}` kan användas för att testa om en variabel är definierad med värde eller inte. (I annat fall hade varje sådant test behövt kontrollera både variabelns existens och om den är tom.)

De två specialstrukturerna för synonymer och hygieniska gränsvärden hanteras så här:

- **Synonymer** hanteras med ett generiskt stöd för tabeller med fyra kolumner, på ett liknande sätt som *nyckelförsedd array*, men med indexering från [0] till [3] för de olika kolumnerna. (Detta medför att kolumn [2] blir ointressant: antingen finns den inte, eller så innehåller den texten ”typ”.)
- **Hygieniska gränsvärden** tillgängliggörs som enskilda värden med namn GRV_*xx*, så att de på ett tydligt sätt kan tas omhand av logiken i Liquid-koden. Dock kan inte parentes användas i variabelnamnet, så gränsvärdena KGV(V)_5 och KGV(V)_15 får variabelnamnen GRV_KGVV_5 och GRV_KGVV_15. Variabelvärdena kompletteras dessutom med enheten ”ppm” (som ju inte ingår i Json-strukturen).

[16] Se https://github.com/sebastienros/fluid för mer information om Fluid.Core, och https://shopify.github.io/liquid/ för mer information om själva skriptspråket.
[17] Även specialvärdet `null` i Json behandlas på detta sätt, d.v.s. variabeln tilldelas inte i Liquid.

5.3.2 Särskilda filter

Templating-motorn Fluid.Core ger möjlighet att definiera egna filter. Vi använder följande specialbyggda filter:

- **imageurl** – Hämtar base64-kodad bild som data-URL för en given bildfil.[18] Jfr kapitel 3.5.
- **imagetext** – Hämtar beskrivningstext för en given bildfil (metadata för *alt*- och *title*-taggarna). Jfr kapitel 3.5.
- **molecule** – Gör om en ren textformel till HTML-kodad textformel med `<sub>`-taggar för siffror som ska vara nedsänkta etc. Se bilaga C.

5.4 Komplettering med HTML-header

Den HTML-kod som genereras när malldokumenten körs genom templating-motorn Fluid.Core är det som ska stå innanför HTML-sidans `<body>`-taggar. I Farliga ämnen-appen kompletterar vi därför koden för varje sida enligt följande:

```
<!doctype html>
<html lang="sv">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="format-detection" content="telephone=no"/>
    <style>
      /* CSS från "css/app.css" */
    </style>
  </head>
  <body>
    <!-- Här infogas koden från templating-motorn -->
  </body>
</html>
```

[18] Vi använder alltså data-URL för att infoga transport- och CLP-etiketter i HTML-sidan, i stället för referenser till fysiska filer. Jfr https://css-tricks.com/data-uris/.

Bilaga A – Hjälpfilen

Hur hjälpsystemet i MSB RIB Farliga ämnen fungerar ingår inte i API-definitionen. I denna bilaga redovisas dock några detaljer om hur den nuvarande implementationen fungerar. Detta kan emellertid ändras utan förvarning.

För närvarande är hela hjälpsystemet integrerat i ett enda dokument: ”help/fahjalp.html”. Det är en komplett HTML-fil inbäddad i Json på detta sätt:

```
{"data":"<!-- HTML-dokument -->"}
```

HTML-filen innehåller utöver HTML-kod även CSS, SVG och Javascript. Vill man använda hjälpfilen behöver man skicka med ett ”#ankarnamn” för att indikera vilken sektion som ska visas:

HTML-ankare	Sektion i hjälpfilen
#sok	Hur sökningen fungerar[19]
#kallor	Var vi hämtar vår data
#f1	Fliken Märkning & ID
#f2	Fliken Räddning
#f3	Fliken Akutvård
#f4	Fliken Fysdata
#f4	Fliken Miljö
#ordlista	Ordlista

Notera också att hjälpfilen endast fungerar korrekt om de typsnitt den hänvisar till finns tillgängliga. Se kapitel 3.8.

[19] Observera att hjälptexten i denna sektion är specifik för MSB:s sökmotor. För utvecklare som väljer att bygga en egen klientapplikation kanske inte texten som står i detta avsnitt stämmer.

Bilaga B – Exempel på komplett ämnespost

Här återges ett verkligt exempel på en komplett ämnespost: Ammoniak, vattenfri (UN 1005). Internt ämnes-id är 448 och ämnesposten lagras i den synkroniserade Farliga ämnen-databasen under nyckelnamnet ”doc/448”.


```
{
  "ID":448,
  "Templates":{
    "Identitet":"id.htm",
    "Raddning":"r2t-m.htm",
    "Akutvard":"sirib45.htm",
    "Fysdata":"fysdata.htm",
    "Miljo":"miljo.htm"
  },
  "data":{
    "CAS":[
      "7664-41-7"
    ],
    "EG":[
      "231-635-3"
    ],
    "Reaktioner":[
      "Vid brand/upphettning bildas nitrösa gaser.",
      "Reagerar häftigt med oxidationsmedel.",
      "Reagerar häftigt med syror.",
      "Vid läckage inomhus kan brännbara koncentrationer uppnås.",
      "Löser sig i vatten och bildar en basisk vattenlösning."
    ],
    "AkutvardInfo":[
      "Verkar även frätande.",
      "Vid brand/upphettning bildas nitrösa gaser."
    ],
    "TEtiketter":[
      "etk_2_3.png",
      "etk_8.png",
      "etk_13_klass2.png"
    ],
    "GhsPiktogram":[
      "ghs04_bottle.png",
      "ghs05_acid.png",
      "ghs06_skull.png",
      "ghs09_aquaticpollut.png"
    ],
    "Anvomr":[
      "Kylmedium.",
      "Framkallare.",
      "Fönsterputsmedel.",
      "Metallytbehandlingsmedel.",
      "Komponent i färg och lack.",
      "Råvara vid plasttillverkning.",
      "Träskyddsmedel."
    ],
    "ToxTolkning":[
      {
        "Akut giftighet":1
      },
      {
        "Anrikas i naturen":2
```

```
  },
  {
    "Cancerframkallande":2
  },
  {
    "Giftigt för akvatiska system":1
  },
  {
    "Hormon/Reproduktionsstörande":3
  },
  {
    "Långsiktiga skador på organ/nervsystem":1
  },
  {
    "Ozonnedbrytande":2
  }
],
"ToxData":[
  {
    "logPow, bioackumulation vid logPow lika med eller över 3.0":
     "0,23 enhetslös"
  },
  {
    "LC50 Daphnia magna, 48 h":"0,66 mg/l"
  },
  {
    "LC50 fisk, 96 h":"0,02 mg/l"
  },
  {
    "EC50 Daphnia magna, 48 h":"25,4 mg/l"
  },
  {
    "IC50 alger, 72 h":"5 mg/l"
  },
  {
    "Giftighet vid inandning: LC50 inhalerat råtta, 4 h":"1,4 mg/l"
  },
  {
    "Giftighet vid inandning: LC50 inhalerat mus, 1 h":"2,9 mg/l"
  },
  {
    "Giftighet vid inandning: LC50 inhalerat kanin, 1 h":"7 mg/l"
  },
  {
    "Giftighet vid förtäring: LD50 oralt råtta":"350 mg/kg"
  }
],
"GhsFaror":[
  {
    "H221":"Brandfarlig gas."
  },
  {
    "H314":"Orsakar allvarliga frätskador på hud och ögon."
  },
  {
    "H331":"Giftigt vid inandning."
  },
  {
    "H400":"Mycket giftigt för vattenlevande organismer."
  }
],
"Synonymer":[
  {
    "en":"Ammonia anhydrous",
```

```
    "typ":"1"
  },
  {
    "en":"Ammonia anhydrous, liquefied"
  },
  {
    "fr":"Ammoniac, anhydre",
    "typ":"1"
  },
  {
    "sv":"Ammoniak"
  },
  {
    "sv":"Ammoniak, vattenfri",
    "typ":"1"
  },
  {
    "sv":"R 717",
    "typ":"2"
  },
  {
    "sv":"R717",
    "typ":"2"
  },
  {
    "de":"Ammoniak, wasserfrei",
    "typ":"1"
  }
],
"Gransvarden":{
  "AEGL3_10":2700.0,
  "AEGL3_30":1600.0,
  "AEGL3_60":1100.0,
  "AEGL3_240":550.0,
  "AEGL3_480":390.0,
  "AEGL2_10":220.0,
  "AEGL2_30":220.0,
  "AEGL2_60":160.0,
  "AEGL2_240":110.0,
  "AEGL2_480":110.0,
  "AEGL1_10":30.0,
  "AEGL1_30":30.0,
  "AEGL1_60":30.0,
  "AEGL1_240":30.0,
  "AEGL1_480":30.0,
  "KGV_5":50.0,
  "NGV_480":20.0,
  "IDLH":300.0,
  "Lukt":50.0,
  "Fornim":5.0
},
"Namn":"Ammoniak, vattenfri",
"Fortydligande":"",
"LosningAvId":"",
"Beskrivning":
 "Giftig och frätande gas (tryckkondenserad). Vid höga koncentrationer inomhus även brandfarlig.",
"FarlNr":"268",
"FarlNrTxt":"Giftig gas, frätande",
"FarlNrADR":"268",
"FgADR":3,
"FgRID":3,
"FgIMDG":3,
"FgIATADGR":4,
"Klass":"2",
```

```
    "Klassif":"2TC",
    "Fpg":"",
    "UN":"1005",
    "Anm":"",
    "AggTill":"Gas; kondenserad",
    "Farg":"Färglös",
    "Lukt":"Skarp; stickande",
    "SmaltP":"-78 °C",
    "KokP":"-33 °C",
    "BrbOmr":"från 15 till 28 vol-%",
    "FlamP":"",
    "TermTP":"630 °C",
    "Dens":"682 kg/m³ vid -33 °C",
    "DensTal":"0,6",
    "Visk":"",
    "VtnLosl":"Lättlöslig i vatten (34 vikt-% vid 20 °C)",
    "Molekyl":"NH3",
    "MolVikt":"17 g/mol",
    "JonPot":"10,18 eV",
    "KritT":"133 °C",
    "AngtryckA":13631304649.645819,
    "AngtryckB":2836.0625548847424,
    "AngtryckPkt":"857 kPa vid 20 °C",
    "Riskomrade":"",
    "Skyddsutrustning":
     "Gastät kemskyddsdräkt (typ 1) och andningsskydd, kompletterad med branddräkt vid brandfara.",
    "Slackmedel":
     "Stoppa i första hand gasflödet. Måste branden släckas, använd tät vattenspray.",
    "FaroangivelseAnmH":"",
    "GhsSignord":"FARA",
    "GhsFinns":"X",
    "GhsAnm":"Ytterligare märkning kan behövas. Se hjälpen."
  }
}
```

Bilaga C – Presentation av molekylformler

I databasen lagras molekylformler som vanlig text, t.ex. ”H2O” för vatten, trots att det korrekta sättet att skriva vattens molekylformel är med nedsänkt tvåa: H_2O. Det är klientapplikationens ansvar att snygga till de råa textmolekylformlerna till snygga, korrekta, molekylformler.

Appen Farliga ämnen använder följande logik för att bearbeta molekylformlerna:

- Alla siffror som står direkt efter en **bokstav**, en **högerparentes** ’)’ eller en **högerklammer** ’]’ visas nedsänkta.
- Efterföljande siffror visas på samma sätt som den inledande siffran, d.v.s. står det C60 så ska både 6 och 0 vara nedsänkta: ”C_{60}”.
- Övriga siffror visas normala, d.v.s. inte nedsänkta: ”U-235”.
- Punkt (.) visas som liten prick i stället (·): ”$CaSO_4 \cdot 2H_2O$”.
- Gement ’x’ behandlas som om det är en siffra: ”Pb_xAsO4”.
- Gement ’n’ behandlas som siffra om det står direkt efter **C**, **H**, en **högerparentes,** en **högerklammer** eller en annan siffra. I övriga fall behandlas det som en bokstav.
- Ett parentesuttryck som enbart innehåller siffror, gement ’n’, plus- eller minustecken, görs om till en nedsänkt formel utan parenteser: (2n+2) blir $_{2n+2}$.

Här följer några exempel för att belysa olika möjliga varianter på molekylformler. De är sorterade efter ökande ”svårighetsgrad”:

Databaslagrad molekylformel	**Molekylformeln presenteras som**
Xe	Xe
In	In
AgCN	AgCN
Li(ClO) (aq)	Li(ClO) (aq)
H2O	H_2O
C88H126O10Mn	$C_{88}H_{126}O_{10}Mn$
Zn3P2	Zn_3P_2
Zr[(NO2)2C6H2(NH2)O]4	$Zr[(NO_2)_2C_6H_2(NH_2)O]_4$
Na3[N(CH2COO)3].1H2O	$Na_3[N(CH_2COO)_3] \cdot 1H_2O$
Na2(NH4)4(V2O7).xH2O	$Na_2(NH_4)_4(V_2O_7) \cdot xH_2O$
(-CH2O-)n	$(-CH_2O-)_n$
CnH(2n+2)	C_nH_{2n+2}

651 81 Karlstad Tel 0771-240 240 **www.msb.se**
Publ.nr MSB1649 - december 2020 ISBN 978-91-7927-087-2