Tutorial


Data Mixer
JSON
Tutorial documentation
Json
XSV
1
30
Auteur: Marcel
Versie 1.0
Mei 2024

Inhoud

Inleiding

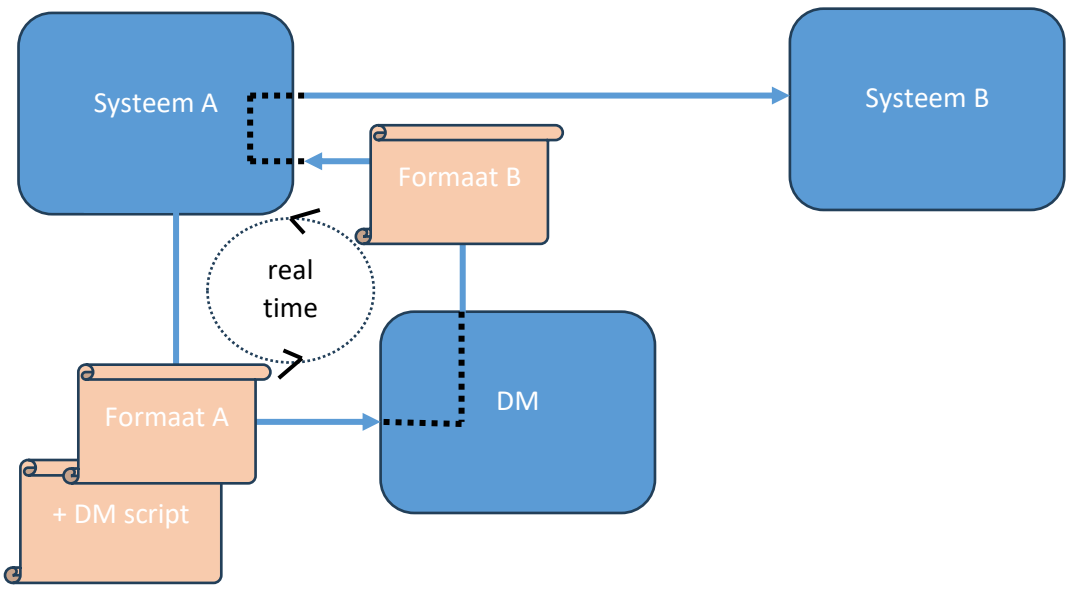
Data Mixer is hét cloud-platform voor het realtime omzetten, converteren en transformeren van gestructureerde digitale data. Het platform van Data Mixer (DM) kan naadloos met uw systeem worden geïntegreerd, waardoor systeem integraties – d.w.z. koppelingen van systeem A naar B en vice versa - eenvoudig mogelijk worden. DM kan zowel door het versturende systeem als het ontvangende systeem worden gebruikt.

Versturende systeem

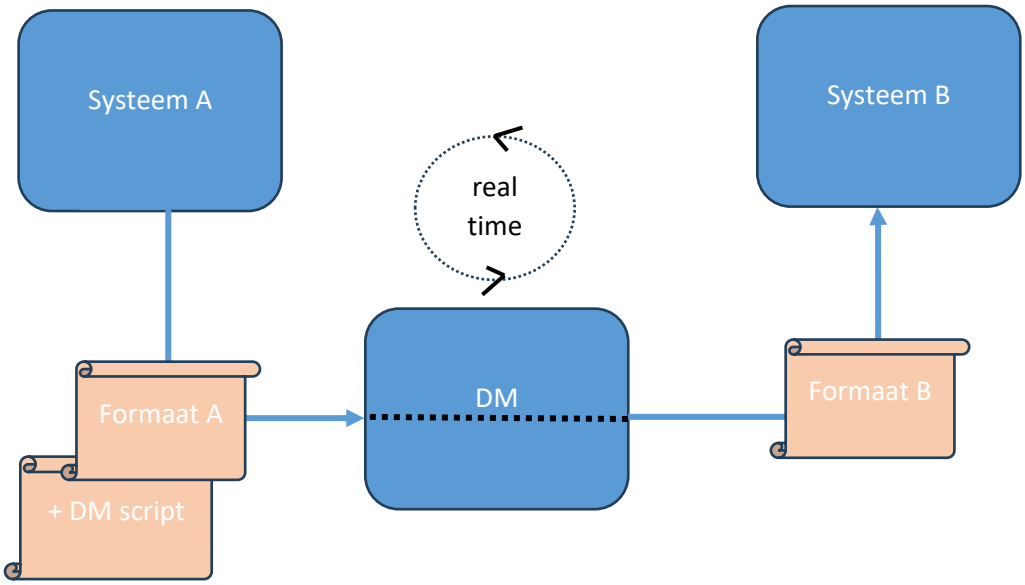
Wanneer het versturende systeem A data in formaat A kan genereren maar in formaat B in het ontvangende systeem B ingelezen moet worden, is het mogelijk om dit via DM te realiseren.

Hierbij zijn de volgende opties mogelijk:

1. Aanroep van DM (met dataformaat A + DM script) waarbij het resultaat (dataformaat B) realtime retour naar systeem A wordt teruggestuurd, en vervolgens vanuit systeem A (direct) naar systeem B wordt (door)gestuurd:



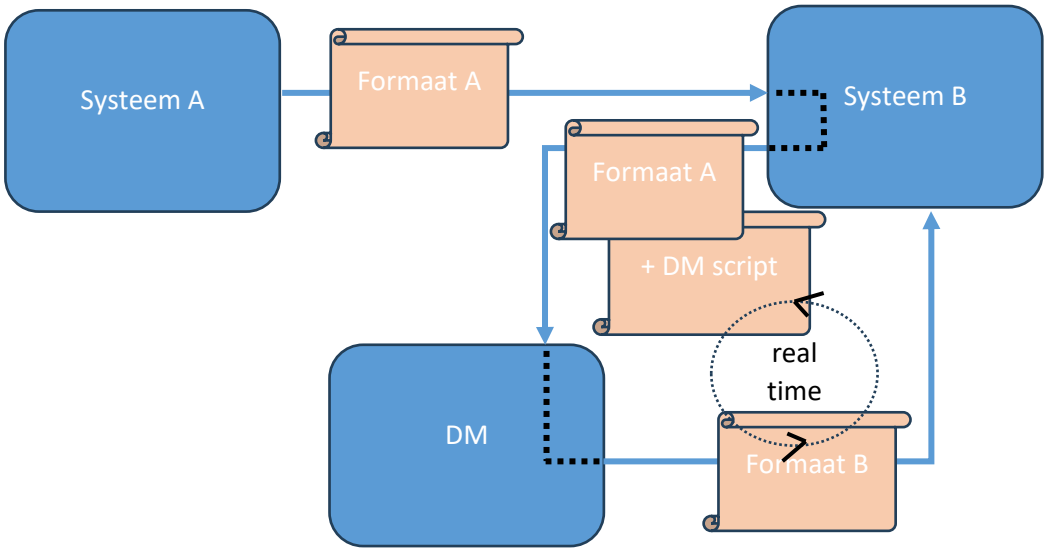
2. Aanroep van DM (met dataformaat A + DM script) waarbij het resultaat (dataformaat B) realtime door DM naar systeem B wordt doorgestuurd:



Hoofdstuk 'Simple MessageBroker' beschrijft deze optie in detail.

Ontvangende systeem

Wanneer het ontvangende systeem B data vanuit een (ander) systeem A in formaat A gestuurd krijgt, maar in formaat B ingelezen moet worden, is het mogelijk om dit door DM te laten doen, door aanroep van DM waarbij formaat B als response wordt ontvangen:



Webservice DM

In dit document wordt uitgelegd hoe DM gebruikt kan worden. De webservice van DM is vrij eenvoudig aan te roepen. Daarbij zijn deze 4 onderdelen zijn nodig:

- Endpoint van DM webservice
- Credentials
- Data
- DM programmeerscript

Endpoint

Het endpoint van DM versie 1.0 is: https://datamixer.nl/api/1.0/

Credentials

Om de webservice te kunnen aanroepen, moet u daartoe een zogenaamde (secret) APIKEY meegeven in de aanroep, waaruit blijkt dat u geautoriseerd bent. Echter, omdat DM 1.0 een free versie betreft, zijn er (nog) geen persoonlijke credentials van toepassing, en is (voor iedere gebruiker) de vaste APIKEY: freeversion.

Data

Met data bedoelen we de brondata in een bepaald gestructureerd formaat, zoals:

- JSON
- XML
- CSV

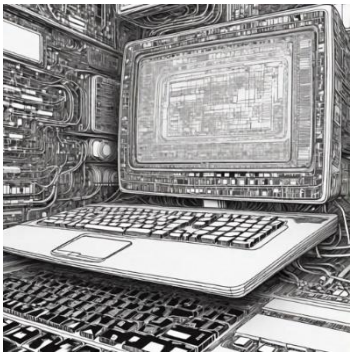
De brondata in één van deze formaten kan worden geconverteerd in een doeldata volgens een door u zelf te bepalen formaat, zoals eveneens:

- JSON
- XML
- CSV

Of een geheel ander formaat, zoals

- YAML
- HTML
- INI
- Eigen (maatwerk) formaten

DM programmeerscript



Met DM programmeerscript (kortweg DM script) is het mogelijk om de brondata te converteren in de doeldata. De DM script bepaalt hoe de doeldata wordt gegenereerd.

DM script is een 'basic' recht toe recht aan programmeertaal, met minimaal een start- en een eind commando, oftewel het DM script bestaat minimaal uit:

```
$start('DataMixer')

$endstart
```

Tussen start en eind wordt het formaat van de doeldata gedefinieerd.

Aanroep DM service

Aanroep van de DM webservice naar het endpoint verloopt met een (http header method) '*POST*' van een JSON **string** met daarin de (bron)data als een array en het DM script als string. De array "data" bestaat hierbij minimaal uit 1 element "content" waarin de brondata (XML, JSON, CSV) als string staat. Er kunnen in de array meerdere (verschillende) bronbestanden worden meegegeven.
Hieronder het eenvoudigste voorbeeld met 1 brondata, het formaat hiervoor is:

```
{"DM":
   {"data": [{content: "….hier staat de brondata…"}],
    "dm$": "… hier staat DM $script om de brondata te converteren …  "
   }
}
```

Het resultaat (Output) is de geconverteerde doeldata die als **string** wordt geretourneerd:

Endpoint	https://datamixer.nl/api/1.0/?apikey=freeversion
APIKEY	freeversion
Input (string)	"{\"DM: ….}"
Output (string)	"Uw doeldata"

DM Programmeer Script: basics

In dit hoofdstuk worden de basis regels van DM Script beschreven.

Met het script tussen het start- en eind commando wordt de doeldata bepaald. Wanneer het gedeelte tussen start en eind leeg is, zoals in onderstaand voorbeeld, dan is de doeldata logischerwijs ook leeg:

Brondata	Don't care
DM script (basic001)	$start('DataMixer') $endstart
Doeldata	""

Zoals in bovenstaande voorbeeld aangegeven bij start en eind begint elke specifieke DM**$**cript commando/functie met het **$**teken: $start en $endstart.

U kunt tussen start en end uw doeldata vormgeven. Daar mag in principe alles staan, d.w.z. teksten komen letterlijk, dus 1-op-1, over in de doeldata. Onderstaande DM script zal, ongeachte welke brondata ook wordt gestuurd, altijd de string "Hello world" opleveren (let op de 2 regelovergangen / carriage returns op regel 1 en regel2 bij de doeldata).

Brondata	Don't care
DM script (basic002)	$start('DataMixer') Hello world $endstart
Doeldata	" Hello world "

Zijn de regelovergangen niet gewenst (carriage returns niet in het resultaat), dan kan de DM script als volgt worden aanpast:

Brondata	Don't care
DM script (basic003)	$start('DataMixer')Hello world$endstart
Doeldata	"Hello world"

Commentaarregels

Extra uitleg over het DM script kan in de vorm van commentaarregels overal geplaats worden, door binnen en/of buiten het start en eind commando de gewenste tekst tussen $/* en $*/ te plaatsen.

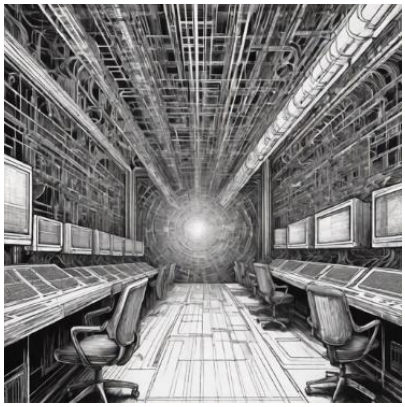
Hieronder 3 voorbeelden van commentaarregels, nr. 1 voor de start, nr. 2 tussen start en einde in, en nr. 3 na het einde. Vanzelfsprekend worden commentaarregels genegeerd en komen **niet** in de doeldata terecht.

Brondata	Don't care
DM script (basic004)	**$/***Commentaar 1: Dit is een voorbeeld van uitleg over het script**$*/** $start('DataMixer')Hello world**$/***Commentaar 2**$*/**$endstart **$/***Commentaar 3**$*/**
Doeldata	"Hello world"

Als alternatief op bovenstaande is het ook mogelijk om de commentaarregels tussen $comment en $endcomment te plaatsen, zoals:

Brondata	Don't care
DM script (basic004)	**$comment**Commentaar 1: Dit is een voorbeeld van uitleg over het script **$endcomment** $start('DataMixer')Hello world$/*Commentaar 2$*/$endstart **$comment** Commentaar 3 **$endcomment**
Doeldata	"Hello world"

Brondata



Tot nu toe is de brondata buiten beschouwing gelaten. Uiteraard zal het de bedoeling zijn om de gegevens uit de brondata erbij te betrekken, zodat ze in een andere vorm als resultaat worden opgeleverd.

De formaten brondata die DM kan converteren zijn, zoals in hoofdstuk 1 beschreven, onder andere JSON, XML en CSV.

In de volgende paragrafen wordt DM script stap voor stap uitgelegd, aan de hand van brondata van een bestand "winkelassortiment / store catalogue" in JSON formaat. Startend met een vrij basaal leeg bestand, eindigen we uiteindelijke in een artikelstructuur en zoomen daar steeds verder op in.

Folderstructuur

Voor het concreet benaderen van de gegevens uit de brondata dient de data als "een soort van" folderstructuur te worden beschouwd.

We starten heel simpel met 2 velden versie (version) en datum van uitgifte (date) in de brondata (in JSON formaat):

```
{"store catalogue":
  {
  "version": 2.3,
  "date": "15-03-2024"
   }
}
```

(basic001)

Door de brondata, in dit geval JSON, als een folderstructuur (zonder driveletters) te beschouwen, komt dat als volgt eruit te zien:

Foldernaam Niveau 0	Foldernaam Niveau 1	Foldernaam Niveau 2
store catalogue		
	version	
	date	

Een waarde van een veld uit de brondata kan worden opgevraagd door dat veld als een folder(structuur) te benaderen. De foldernamen worden gescheiden door de (forward)slash "/", **en** begint met de dubbele (forward)slash "//"

Bijvoorbeeld: het uitlezen van het gegeven "version" vindt plaats via het "folderpad":

```
//store catalogue/version
```

De algemene syntax van een folderexpressie is:

```
//foldernaam1/foldernaam2/foldernaam3
```

Let op: de foldernamen in de folderexpressie zijn **case-sensitive** wat betekent dat de letters exact in hoofdletters en kleine letters moet overeenkomen met de brondata!

Verder is het niet perse nodig om bij foldernaam1 te beginnen, m.a.w. het is ook mogelijk om ineens bij foldernaam2 of foldernaam3 (of nog dieper) te starten. Echter telkens begint zo'n folderpad met '//', b.v.

```
//foldernaam2/foldernaam3
```

of

```
//foldernaam3
```

zijn eveneens toegestane folders waarbij de voorgaande folders 'geskipt' kunnen worden.

In een folderexpressie kan bij elke genoemde foldernaam aanvullend een voorwaarde (conditie) worden meegegeven, om het juiste folderpad te benaderen. Dat gebeurt via de $filter-functie. Uitleg volgt later, o.a. in de $loop-functie komt dit aan de orde.

De syntax is dan:

```
//foldernaam1$filter(conditieexpressie)/foldernaam2$filter(conditieexpressie)
```

De uitleg van conditieexpressies komt ook later aan de orde, hier al wel een voorbeeld voor de beeldvorming, waarbij de folder version met waarde 2.3 wordt benaderd (en met andere waarden niet):

```
//store catalogue$filter(version=2.3)/version
```

Bijzonderheid: In plaats van een conditionele expressie in de functie $filter is het ook mogelijk om geen conditie, maar direct een positief getal (integer) in te vullen (>0), waarmee de in geval van meerdere dezelfde foldernamen gericht op een "foldernummer" kan worden gewezen. D.w.z. de hoeveelste foldernaam, van boven naar beneden in de brondata bekeken, moet worden benaderd. In JSON kan dit een array van waarden zijn, maar ook bij 1 enkele waarde (zoals version) kan dit gebruikt worden. In XML kan dit deel-XML met dezelfde tagnamen zijn, en in CSV is dit over het algemeen een lijst met meerdere rijen (waarbij het nodig is om de 3e rij uit te lezen, bijvoorbeeld).

```
//store catalogue/version$filter(1)
```

Tenslotte kan een folderpad betrekking hebben op een ARRAY van elementen, zoals eigenlijk alleen bij JSON voorkomt, zoals bijvoorbeeld:

```
{"element-array":
   [
    {"name": "abc"},
    {"name": "def"},
    {"name": "ghi"}
   ]
}
```

In het geval dat de foldernaam een array is, dan is het verplicht om de $array functie in de folderexpressie te gebruiken:

```
$array(//element-array)$filter(2)/name levert "def" op
```

En

```
$array(//element-array)$filter(name='ghi')/name levert "ghi" op
```

$value

Met de functie $value wordt de waarde van een veld uit de brondata uitgelezen:

Functie	**$value**
Effect	Uitlezen van een (waarde van een) veld in de brondata om in de doeldata te verwerken
Input	Folderexpressie
Aanroep	$value (folderexpressie)
Output	Waarde uit de brondata
Voorbeeld	**$value**(//store catalogue/version) levert 2.3 op. **$value**(//store catalogue$filter(version=2.3)/version) levert 2.3 op. **$value**(//store catalogue$filter(version=2.4)/version) levert niets op (lege waarde). **$value**(//store catalogue/version$filter(.=2.3)) levert 2.3 op. **$value**(//store catalogue/version$filter(.=2.4)) levert niets op (lege waarde)

Voorbeeld:

De brondata "store catalogue" uit formaat A moet in formaat B "Shop Pricelist" worden genoemd. De versie "version" (die we in formaat B "Version" met hoofdletter noemen) lezen we uit, en dat moet geen type "number" maar "string" zijn, omdat de prefix "V" in ervoor dient te staan:

Brondata
(basic001)
	{"store catalogue":
	{
	"version": 2.3,
	"date": "15-03-2024"
	}
	}

DM script (basic005)	$start('DataMixer') {"Shop Pricelist": {"Version": "V**$value**(//store catalogue/version)"} } $endstart
Doeldata	{"Shop Pricelist": {**"Version": "V2.3"**} }

Bijzonderheden:

Wanneer er geen folderpad wordt meegegeven, dus "**$value**()", dan geeft dat een error.

Bij een niet-bestaand folderpad wordt geen error gegeven, maar wordt geen resultaat teruggegeven. Zoals reeds genoemd is een folderpad case-sensitive: het DM-script "**$value**(//Store Catalogue/version)" levert derhalve geen waarde op, omdat dit folderpad met hoofdletter "S" (van Store) niet herkend wordt (had store moeten zijn). Een geheel ander folderpad, b.v. "**$value**(//IdontExist/version)" levert eveneens geen waarde op.

Folderpaden zijn relatief, dus b.v. **$value**(//store catalogue/version/../version) is ook toegestaan (en levert eveneens 2.3 op).

Er kan met $value in plaats van het opgeven van een folderpad ook direct gerekend worden: b.v.:

$value(2+3) levert 5 op,

$value(2*3) levert 6 op

$value(2-3) levert -1 op

Het rekenen wordt vaak toegepast in combinatie met gegevens uit het bronbestand. Hier komen we later uitgebreid in detail op terug.

DM Programmeer Script: stringfuncties

In dit hoofdstuk worden de string-functies behandeld.

$substr

De functie $substr kan worden toegepast op een string-waarde uit de brondata om daarmee een gedeelte van een string uit te lezen:

Functie	**$substr**
Effect	Uitlezen van een deel van een string in de brondata
Input	1. Tekst (string) 2. Startpositie (integer), minimaal 1. 3. Lengte (integer)
Aanroep	$substr (folderexpressie, startpos, lengte)
Output	Waarde uit de brondata, vanaf startpositie met opgegeven lengte
Voorbeeld	$substr(//store catalogue/date, 7,4) levert 2024 op

Voorbeeld 1:

De datum (dd-mm-yyyy) uit de brondata gaan we opsplitsen in 3 velden: jaar, maand en dag. Hiertoe gebruiken we $substr om de 3 delen van de datum (date) uit te lezen:

Pos.	1	2	3	4	5	6	7	8	9	10
Teken	1	5	-	0	3	-	2	0	2	4

Het jaar (folderexpressie = //store catalogue/date) begint op startpositie 7 en heeft een vaste lengte van 4 posities. De maand begint op startpositie 4 en heeft een vaste lengte van 2. De dag begint vooraan op positie 1 en heeft lengte 2:

Brondata
(basic001)
	{"store catalogue":
	{
	"version": 2.3,
	"date": "15-03-2024"
	}

DM script (basic006)	$start('DataMixer') {"Shop Pricelist": {"Version": "V$value(//store catalogue/version)", "Year": $value(**$substr**(//store catalogue/date, 7,4)), "Month": $value(**$substr**(//store catalogue/date, 4,2)), "Day": $value(**$substr**(//store catalogue/date, 1,2)) } }

	$endstart
Doeldata	{"Shop Pricelist": {"Version": "V2.3", **"Year": 2024,** **"Month": 03,** **"Day": 15** } }

Voorbeeld 2:

Van de datum in formaat dd-mm-yyyy uit de brondata is het de bedoeling om deze omgekeerd in formaat yyyy-mm-dd in de doeldata op te leveren. Hiertoe gebruiken we eveneens $substr om de 3 delen van de datum (date) uit te lezen en in 1 veld Date op te leveren:

Brondata
(basic001)
	{"store catalogue":
	{
	"version": 2.3,
	"date": "15-03-2024"
	}

DM script (basic007)	$start('DataMixer') {"Shop Pricelist": {"Version": "V$value(//store catalogue/version)", "Date": "$value(**$substr**(//store catalogue/date, 7,4))-$value(**$substr**(//store catalogue/date, 4,2))-$value(**$substr**(//store catalogue/date,1,2))" } } $endstart
Doeldata	{"Shop Pricelist": {"Version": "V2.3", **"Date": "2024-03-15"** } }

Bijzonderheden:

Bij een niet-bestaande folderpad levert $substr geen resultaat op (lege waarde).

Een startpositie kleiner dan 1 (0 of negatief) is wel mogelijk maar weinig zinvol. In dat geval levert $substr alleen resultaat op als de lengte groter is dan 2 + absolute waarde van startpositie.

Een startpositie groter dan de lengte van de (bron)string levert altijd de lege string op.

Een lengte van 0 of kleiner levert eveneens altijd de lege string op. Een lengte die groter is dan de lengte van de (bon)string is toegestaan, maar weinig zinvol.

$strcat

De functie $strcat kan worden toegepast om teksten (deelstringen) achter elkaar te plakken (concateneren):

Functie	**$strcat**
Effect	Het concateneren van meerdere teksten (strings)
Input	Max. 5 teksten (strings)
Aanroep	$strcatr (folderexpressie of string, folderexpressie of string, folderexpressie of string, folderexpressie of string, folderexpressie of string)
Output	De geconcateneerde tekst
Voorbeeld	$strcat('V', //store catalogue/version) levert V2.3 op

Voorbeeld :

Van de datum in formaat dd-mm-yyyy uit de brondata is het de bedoeling om deze omgekeerd in formaat yyyy-mm-dd in de doeldata op te leveren. Hiertoe gebruiken we eveneens $strcat om de 3 delen van de datum (date) in 1 veld Date op te leveren:

Brondata
(basic001)
	{"store catalogue":
	{
	"version": 2.3,
	"date": "15-03-2024"
	}

DM script (basic008)	$start('DataMixer') {"Shop Pricelist": {"Version": "V$value(//store catalogue/version)", "Date": "$value**($strcat**($substr(//store catalogue/date, 7, 4),'-', $substr(//store catalogue/date, 4,2), '-', $substr(//store catalogue/date,1,2)))" } } $endstart
Doeldata	{"Shop Pricelist": {"Version": "V2.3", **"Date": "2024-03-15"** } }

$strlen

De functie $strlen wordt gebruikt om de lengte van een string te bepalen:

Functie	**$strlen**
Effect	De lengte van het aantal tekens van een tekst (string)
Input	Tekst (string)
Aanroep	$strlen (folderexpressie)
Output	Lengte van de tekst (integer)
Voorbeeld	$strlen(//store catalogue/date) levert 10 op

Voorbeeld :

De lengte van de versie inclusief de 'V' wordt in een apart veld 'VersionLength' bewaard:

Brondata
(basic001)
	{"store catalogue":
	{
	"version": 2.3,
	"date": "15-03-2024"
	}

DM script (basic009)	$start('DataMixer') {"Shop Pricelist": {"Version": "V$value(//store catalogue/version)", "VersionLength": $value(**$strlen**($strcat('V', //store catalogue/version))) } } $endstart
Doeldata	{"Shop Pricelist": {"Version": "V2.3", **"VersionLength": 4** } }

$substr_before en $substr_after

De functie $substr_before en $substr_after worden gebruikt om een gedeelte van een tekst (string) resp. voor en na een bepaald teken of tekst te bepalen:

Functie	**$substr_before**
Effect	Uitlezen van een deel van een string in de brondata voor een bepaalde teken of tekst
Input	1. Tekst1 (string) 2. Tekst2 (string), minimaal 1 teken.
Aanroep	$substr_before (folderexpressie, char_to_search_for)
Output	Het gedeelte van tekst1 van de brondata dat zich voor tekst2 van de brondata bevindt
Voorbeeld	$substr_before (//store catalogue/date, '-') levert 15 op

Functie	**$substr_after**
Effect	Uitlezen van een deel van een string in de brondata na een bepaalde teken of tekst
Input	1. Tekst1 (string) 2. Tekst2 (string), minimaal 1 teken.
Aanroep	$substr_after (folderexpressie, char_to_search_for)
Output	Het gedeelte van tekst1 van de brondata dat zich na tekst2 van de brondata bevindt
Voorbeeld	$substr_after (//store catalogue/date, '-') levert 03-2024 op

Voorbeeld :

De versie wordt gesplitst in de hoofdversie (het gedeelte voor de punt) en de subversie (het gedeelte achter de punt). Verder wordt dag, maand en jaar van de datum (date) bepaald via $substr_before en $substr_after a.d.h.v. zoeken op het teken '-':

Brondata
(basic001)
	{"store catalogue":
	{
	"version": 2.3,
	"date": "15-03-2024"
	}

DM script (basic010)	$start('DataMixer') {"Shop Pricelist": {"MajorVersion": $value(**$substr_before**(//store catalogue/version,'.')), "MinorVersion": $value(**$substr_after**(//store catalogue/version,'.')), "Year": $value(**$substr_after($substr_after**(//store catalogue/date, '-'),'-')), "Month: $value(**$substr_before($substr_after**(//store catalogue/date, '-'),'-')), "Day": $value(**$substr_before**(//store catalogue/date, '-'))

	} } $endstart
Doeldata	{"Shop Pricelist": {"MajorVersion": 2, "MinorVersion": 3, "Year": 2024, "Month: 03, "Day": 15 } }

$replace

De functie $replace wordt gebruikt om een teken in een string te wijzigen in een ander teken:

Functie	**$replace**
Effect	Tekstaanpassing van character/teken 'x' in 'y'
Input	Brontekst (string), aan te passen character (string), aanpassen in character (string)
Aanroep	$replace(folderexpressie, char_to_change, char_to_change_in)
Output	Aangepaste tekst (string)
Voorbeeld	$replace(//store catalogue/date, '-','/') levert "15/03/2024" op

Voorbeeld :

In de datum worden de steepjes '-' vervangen door een slash "/", en de punt in de versie halen we weg (vervangen door lege string):

Brondata
(basic002)
	{"store catalogue":
	{
	"name": "Greengrocer supermarket Ltd",
	"version": 2.3,
	"date": "15-03-2024"
	}
	}

DM script (basic011)	$start('DataMixer') {"Shop Pricelist": {"Version": "V$value(**$replace**(//store catalogue/version,'.',''))", "Date": "$value(**$replace**(//store catalogue/date, '-', '/'))" } } $endstart
Doeldata	{"Shop Pricelist": {**"Version": "V23"**, **"Date": "15/03/2024"** } }

Bijzonderheden:

De replace functie werkt voor één character (teken), en wordt vervangen door één (ander) character (teken). M.a.w. de replace vervangt **niet meerdere tekens (substring) in één keer**, maar vervangt teken voor teken. Zo is het resultaat van:

$replace(//store catalogue/name, 'Ltd','bv') niet "Greengrocer supermarket bv", maar worden alle hoofdletters 'L' vervangen door 'b', de letter 't' door 'b' en de 'd'

door de lege string, waardoor als resultaat "Greengrocer supermarke**v** bv" wordt verkregen (wat wellicht niet geheel de bedoeling is).

Verder is de replace functie **case sensitive / hoofdletter gevoelig**:

$replace(//store catalogue/name, 'E','x'), zal alle hoofdletters 'E' vervangen in de kleine letter 'x', en omdat "Greengrocer supermarket Ltd" geen "E" bevat, is het resultaat dat er niets wordt aangepast, en de oorspronkelijke tekst wordt opgeleverd: "Greengrocer supermarket Ltd".

En:

$replace(//store catalogue/name, 'Gg','gG'), zal alle hoofdletters 'G' vervangen in de kleine letter 'g' en omgekeerd, waardoor de tekst "**g**reen**G**rocer supermarket Ltd" wordt opgeleverd.

$findstr

De functie $findstr wordt gebruikt om te controleren of een teken of substring in een tekststring aanwezig is:

Functie	**$findstr**
Effect	Controleert of een (of meer) character(s)/ teken(s) in een tekst voorkomt
Input	Tekst om te controleren, zoektekst (string)
Aanroep	$findstr (folderexpressie, zoektekst)
Output	True of false voor resp. wel en niet gevonden
Voorbeeld	$findstr(//store catalogue/name, 'Ltd') levert true op

Voorbeeld :

Controle of de term Ltd in de naam van de store catalogue voorkomt:

Brondata
(basic002)
	{"store catalogue":
	{
	"name": "Greengrocer supermarket Ltd",
	"version": 2.3,
	"date": "15-03-2024"
	}
	}

DM script (basic012)	$start('DataMixer') {"Shop Pricelist": {"Ltd in name": $value(**$findstr**(//store catalogue/name,'Ltd')), "Version": "V$value($replace(//store catalogue/version,'.',''))", "Date": "$value($replace(//store catalogue/date, '-', '/'))" } } $endstart
Doeldata	{"Shop Pricelist": {**"Ltd in name": true**, "Version": "V23", "Date": "15/03/2024" } }

Bijzonderheden:

De functie $findstr is **case sensitive / hoofdletter gevoelig**:

$findstr(//store catalogue/name, 'E') levert false op omdat de hoofdletters 'E' niet bestaat, echter $findstr(//store catalogue/name, 'e') levert true op.

$upper

De functie $upper wordt gebruikt om alle characters (tekens) in een tekst om te zetten in hoofdletters:

Functie	**$upper**
Effect	De tekst (string) wordt omgezet in hoofdletters
Input	tekst (string)
Aanroep	$upper (folderexpressie)
Output	Tekst (string)
Voorbeeld	$upper(//store catalogue/name) levert "GREENGROCER SUPERMARKET LTD" op

Voorbeeld :

De naam van de supermarkt wordt omgezet in hoofdletters:

Brondata
(basic002)
	{"store catalogue":
	{
	"name": "Greengrocer supermarket Ltd",
	"version": 2.3,
	"date": "15-03-2024"
	}
	}

DM script (basic013)	$start('DataMixer') {"Shop Pricelist": {"Name": "$value(**$upper**(//store catalogue/name))", "Version": "V$value($replace(//store catalogue/version,'.',''))", "Date": "$value($replace(//store catalogue/date, '-', '/'))" } } $endstart
Doeldata	{"Shop Pricelist": {**"Name": "GREENGROCER SUPERMARKET LTD"**, "Version": "V23", "Date": "15/03/2024" } }

$lower

De functie $lower wordt gebruikt om alle characters (tekens) in een tekst om te zetten in kleine letters:

Functie	**$lower**
Effect	De tekst (string) wordt omgezet in kleine letters
Input	tekst (string)
Aanroep	$lower (folderexpressie)
Output	Tekst (string)
Voorbeeld	$lower(//store catalogue/name) levert "greengrocer supermarket ltd" op

Voorbeeld :

De naam van de supermarkt wordt omgezet in kleine letters:

Brondata
(basic002)
	{"store catalogue":
	{
	"name": "Greengrocer supermarket Ltd",
	"version": 2.3,
	"date": "15-03-2024"
	}
	}

DM script (basic014)	$start('DataMixer') {"Shop Pricelist": {"Name": "$value(**$lower**(//store catalogue/name))", "Version": "V$value($replace(//store catalogue/version,'.',''))", "Date": "$value($replace(//store catalogue/date, '-', '/'))" } } $endstart
Doeldata	{"Shop Pricelist": {**"Name": "greengrocer supermarket ltd"**, "Version": "V23", "Date": "15/03/2024" } }

DM Programmeer Script: rekenfuncties

$random

Met de functie $random wordt een willekeurig getal tussen 0 en 1 gegenereerd:

Functie	**$random**
Effect	
Input	-
Aanroep	$random
Output	Getal tussen 0 en 1
Voorbeeld	$random levert b.v. 0.472316639283741 op

Voorbeeld :

Er wordt een uniek getal in het veld "Willekeurig getal" (real) bepaald:

Brondata
(basic002)
	{"store catalogue":
	{
	"name": "Greengrocer supermarket Ltd",
	"version": 2.3,
	"date": "15-03-2024"
	}
	}

DM script (basic017)	$start('DataMixer') {"Shop Pricelist": {"Random number": $value(**$random**), "Name": "$value($lower(//store catalogue/name))", "Version": "V$rpad(//store catalogue/version,'000',5)", "Date": "$value($replace(//store catalogue/date, '-', '/'))" } } $endstart
Doeldata	{"Shop Pricelist": {**"Random number": 0.8573740581337015**, "Name": "greengrocer supermarket ltd", "Version": "V2.300", "Date": "15/03/2024" } }

*

Met de functie * vindt een vermenigvuldiging tussen 2 of meer getallen plaats.

Functie	*
Effect	
Input	Getallen
Aanroep	*
Output	Resultaat van een vermenigvuldiging
Voorbeeld	2 * 3 levert 6 op, 2 * 3* 4 levert 24 op

Voorbeeld :

Er wordt een uniek getal in het veld "Willekeurig getal" (real) bepaald, tussen 0 en 10:

Brondata
(basic002)
	{"store catalogue":
	{
	"name": "Greengrocer supermarket Ltd",
	"version": 2.3,
	"date": "15-03-2024"
	}
	}

DM script (basic018)	$start('DataMixer') {"Shop Pricelist": {"Random number": $value($random * 10), "Name": "$value($lower(//store catalogue/name))", "Version": "V$rpad(//store catalogue/version,'000',5)", "Date": "$value($replace(//store catalogue/date, '-', '/'))" } } $endstart
Doeldata	{"Shop Pricelist": {**"Random number": 3.6090399615520936**, "Name": "greengrocer supermarket ltd", "Version": "V2.300", "Date": "15/03/2024" } }

+

Met de functie + vindt een optelling tussen 2 of meer getallen plaats.

Functie	+
Effect	
Input	Getallen
Aanroep	+
Output	Resultaat van een optelling (som van getallen)
Voorbeeld	2 + 3 levert 5 op, 2 + 3 + 4 levert 9 op

Voorbeeld :

Er wordt een uniek getal in het veld "Willekeurig getal" (real) bepaald, tussen 5 en 15:

Brondata
(basic002)
	{"store catalogue":
	{
	"name": "Greengrocer supermarket Ltd",
	"version": 2.3,
	"date": "15-03-2024"
	}
	}

DM script (basic019)	$start('DataMixer') {"Shop Pricelist": {"Random number": $value($random * 10 + 5), "Name": "$value($lower(//store catalogue/name))", "Version": "V$rpad(//store catalogue/version,'000',5)", "Date": "$value($replace(//store catalogue/date, '-', '/'))" } } $endstart
Doeldata	{"Shop Pricelist": {**"Random number": 11.18805798420105**, "Name": "greengrocer supermarket ltd", "Version": "V2.300", "Date": "15/03/2024" } }

Met de functie – wordt het verschil tussen 2 of meer getallen berekend.

Functie	-
Effect	
Input	Getallen
Aanroep	-
Output	Resultaat van een optelling (som van getallen)
Voorbeeld	2 - 3 levert -1 op, 2 - 3 - 4 levert -5 op

Voorbeeld :

Er wordt een uniek getal in het veld "Willekeurig getal" (real) bepaald, tussen -5 en 5:

Brondata
(basic002)
	{"store catalogue":
	{
	"name": "Greengrocer supermarket Ltd",
	"version": 2.3,
	"date": "15-03-2024"
	}
	}

DM script (basic020)	$start('DataMixer') {"Shop Pricelist": {"Random number": $value($random * 10 - 5), "Name": "$value($lower(//store catalogue/name))", "Version": "V$rpad(//store catalogue/version,'000',5)", "Date": "$value($replace(//store catalogue/date, '-', '/'))" } } $endstart
Doeldata	{"Shop Pricelist": {**"Random number": -4.102763869330058**, "Name": "greengrocer supermarket ltd", "Version": "V2.300", "Date": "15/03/2024" } }

$round

Met de functie $round wordt een niet geheel getal (real) afgerond naar een geheel getal (integer), naar boven of naar onder, afhankelijk van de decimaal.

Functie	**$round**
Effect	Afronden van een getal
Input	Getal
Aanroep	$round(getal)
Output	Getal wordt naar beneden afgerond indien decimaal <.5 is, en naar boven afgerond indien decimaal >=.5
Voorbeeld	$round(1.49) levert 1 op, $round(1.5) levert 2 op.

Voorbeeld :

Er wordt een uniek getal in het veld "Willekeurig getal" (integer) bepaald, tussen 0 en 10:

Brondata
(basic002)
	{"store catalogue":
	{
	"name": "Greengrocer supermarket Ltd",
	"version": 2.3,
	"date": "15-03-2024"
	}
	}

DM script (basic021)	$start('DataMixer') {"Shop Pricelist": {"Random number": $value(**$round**($random * 10)), "Name": "$value($lower(//store catalogue/name))", "Version": "V$rpad(//store catalogue/version,'000',5)", "Date": "$value($replace(//store catalogue/date, '-', '/'))" } } $endstart
Doeldata	{"Shop Pricelist": {**"Random number": 4,** "Name": "greengrocer supermarket ltd", "Version": "V2.300", "Date": "15/03/2024" } }

$div

Met de functie $div wordt een getal (real) gedeeld door een ander getal

Functie	**$div**
Effect	Het delen van getal door een deelgetal
Input	Getal
Aanroep	$div(getal, deelgetal)
Output	Getal gedeeld door deelgetal wordt onafgerond opgeleverd
Voorbeeld	$div(2,3) levert 1 op, $round(1.5) levert 2 op.

Voorbeeld :

Er wordt een uniek getal in het veld "Willekeurig getal" (integer of real met .5) bepaald, tussen 0 en 5 (dus 0, 0.5, 1, 1.5, 2, …, 4.5 of 5), hiervoor is een deling door 2 nodig:

Brondata
(basic002)
	{"store catalogue":
	{
	"name": "Greengrocer supermarket Ltd",
	"version": 2.3,
	"date": "15-03-2024"
	}
	}

DM script (basic022)	$start('DataMixer') {"Shop Pricelist": {"Random number": $value(**$div**($round($random * 10),2)), "Name": "$value($lower(//store catalogue/name))", "Version": "V$rpad(//store catalogue/version,'000',5)", "Date": "$value($replace(//store catalogue/date, '-', '/'))" } } $endstart
Doeldata	{"Shop Pricelist": {**"Random number": 3.5**, "Name": "greengrocer supermarket ltd", "Version": "V2.300", "Date": "15/03/2024" } }

Opmerking: delen door 0 is wiskundig gezien niet mogelijk, het deelgetal mag in principe geen 0 zijn. Wanneer $div toch met deelgetal 0 wordt aangeroepen, is het resultaat 'Infinity'.

$floor

Met de functie $floor wordt een niet geheel getal (real) naar beneden afgerond naar een geheel getal (integer), onafhankelijk van de decimaal.

Functie	**$floor**
Effect	Afronden van een getal naar beneden
Input	Getal
Aanroep	$floor(getal)
Output	Getal wordt naar beneden afgerond
Voorbeeld	$round(1.49) levert 1 op, $round(1.5) levert 1 op, $round(1.999) levert 1 op

Voorbeeld :

Er wordt een uniek getal in het veld "Willekeurig getal" (integer) bepaald, tussen 0 en 9:

Brondata
(basic002)
	{"store catalogue":
	{
	"name": "Greengrocer supermarket Ltd",
	"version": 2.3,
	"date": "15-03-2024"
	}
	}

DM script (basic023)	$start('DataMixer') {"Shop Pricelist": {"Random number": $value(**$floor**($random * 10)), "Name": "$value($lower(//store catalogue/name))", "Version": "V$rpad(//store catalogue/version,'000',5)", "Date": "$value($replace(//store catalogue/date, '-', '/'))" } } $endstart
Doeldata	{"Shop Pricelist": {**"Random number": 4,** "Name": "greengrocer supermarket ltd", "Version": "V2.300", "Date": "15/03/2024" } }

$ceil

Met de functie $ceil wordt een niet geheel getal (real) naar boven afgerond naar een geheel getal (integer), onafhankelijk van de decimaal.

Functie	**$ceil**
Effect	Afronden van een getal naar boven
Input	Getal
Aanroep	$ceil(getal)
Output	Getal wordt naar boven afgerond
Voorbeeld	$round(1.49) levert 2 op, $round(1.5) levert 2 op, $round(2.001) levert 3 op

Voorbeeld :

Er wordt een uniek getal in het veld "Willekeurig getal" (integer) bepaald, tussen 1 en 10:

Brondata
(basic002)
	{"store catalogue":
	{
	"name": "Greengrocer supermarket Ltd",
	"version": 2.3,
	"date": "15-03-2024"
	}
	}

DM script (basic024)	$start('DataMixer') {"Shop Pricelist": {"Random number": $value(**$ceil**($random * 10)), "Name": "$value($lower(//store catalogue/name))", "Version": "V$rpad(//store catalogue/version,'000',5)", "Date": "$value($replace(//store catalogue/date, '-', '/'))" } } $endstart
Doeldata	{"Shop Pricelist": {**"Random number": 4,** "Name": "greengrocer supermarket ltd", "Version": "V2.300", "Date": "15/03/2024" } }

$count

Met de functie $count kan worden bepaald hoeveel keren een bepaald gegeven in (een deel van) de data voorkomt

Functie	**$count**
Effect	Tellen hoe vaak de folderexpressie voorkomt
Input	Folderexpressie
Aanroep	$count(folderexpressie)
Output	Het aantal keer dat de folderexpressie voorkomt in de data
Voorbeeld	$count(//store catalogue/name) levert 1 op, $count(//name) levert 4 op

Voorbeeld :

Het aantal fruitartikelen in de catalogus wordt bepaald(*):

Brondata
(basic003)
	{"store catalogue":
	{
	"name": "Greengrocer supermarket Ltd",
	"version": 2.3,
	"date": "15-03-2024",
	"fruit":
	[
	{
	"name": "apple",
	"color": "red",
	"weight": 0.26
	},
	{
	"name": "banana",
	"color": "yellow",
	"weight": 0.18
	},
	{
	"name": "strawberry",
	"color": "red",
	"weight": 0.03
	}
	]
	}
	}

DM script (basic025)	$start('DataMixer') {"Shop Pricelist": {"Random number": $value($ceil($random * 10)), "Name": "$value($lower(//store catalogue/name))", "Version": "V$rpad(//store catalogue/version,'000',5)", "Date": "$value($replace(//store catalogue/date, '-', '/'))",

	"Number of fruits": $value(**$count**(//name) - **$count**(//store catalogue/name)) } } $endstart
Doeldata	{"Shop Pricelist": {"Random number": 8, "Name": "greengrocer supermarket ltd", "Version": "V2.300", "Date": "15/03/2024", **"Number of fruits": 3** } }

(*) Let op dat de methode in dit voorbeeld prima werkt, maar niet robuust en sterk afhankelijk is van de data is, wordt ergens nog een “name” in de toekomst gebruikt, dan klopt dit resultaat mogelijk niet meer. Er bestaat een beter methode via $array, zie verderop in dit document.

$sum

Met de functie $sum worden gegevens in (een deel van) de data bij elkaar opgeteld

Functie	**$sum**
Effect	Optelling van getalwaarden die voldoen aan de folderexpressie.
Input	Folderexpressie
Aanroep	$sum(folderexpressie)
Output	De som van alle gegevens (getallen) die via folderexpressie wordt gevonden.
Voorbeeld	$sum(//weight) levert

Voorbeeld :

Het gewicht van elk 1 fruitartikel getotaliseerd/opgeteld wordt als volgt bepaald:

Brondata
(basic003)
	{"store catalogue":
	{
	"name": "Greengrocer supermarket Ltd",
	"version": 2.3,
	"date": "15-03-2024",
	"fruit":
	[
	{
	"name": "apple",
	"color": "red",
	"weight": 0.26
	},
	{
	"name": "banana",
	"color": "yellow",
	"weight": 0.18
	},
	{
	"name": "strawberry",
	"color": "red",
	"weight": 0.03
	}
	]
	}
	}

DM script (basic026)	$start('DataMixer') {"Shop Pricelist": {"Random number": $value($ceil($random * 10)), "Name": "$value($lower(//store catalogue/name))", "Version": "V$rpad(//store catalogue/version,'000',5)", "Date": "$value($replace(//store catalogue/date, '-', '/'))",

	"Number of fruits": $value($count(//name) - $count(//store catalogue/name)), "Total weight of fruit": $value**($sum**(//weight) } } $endstart
Doeldata	{"Shop Pricelist": {"Random number": 2, "Name": "greengrocer supermarket ltd", "Version": "V2.300", "Date": "15/03/2024", "Number of fruits": 3, **"Total weight of fruit": 0.47** } }

DM Programmeer Script: datumfuncties

Er zijn twee datums functies om de systeemdatum (datum van vandaag) op te vragen.

$localdate

Met de functie (of beter gezegd globale) $localdate wordt de huidige datum en tijd (systeemdatum) teruggeven:

Functie	**$localdate**
Effect	Bepalen van de huidige systeemdatum & tijd
Input	-
Aanroep	$localdate let op, geen haakjes, het is feitelijk een globale variabele
Output	Datum+tijd in formaat YYYY-MM-DDThh:mi:ss
Voorbeeld	$localdate levert 18-05-2024T10:49:32 op

Voorbeeld :

We gaan het veld "Willekeurig getal" aanpassen in "MessageId" zodat elke te versturen bericht een unieke ID meekrijgt. Daartoe gebruiken we $localdate om al vrij uniek te zijn, en verwijderen via $replace de tussenliggende tekens als 'T', '-' en ':'. Samen (via $strcat) met een $random nummer tot 10 cijfers (vandaar de vermenigvuldiging met 10) levert dat op elk moment een uniek ID op:

Brondata (basic003)	{"store catalogue": { "name": "Greengrocer supermarket Ltd", "version": 2.3, "date": "15-03-2024", "fruit": [{ "name": "apple", "color": "red", "weight": 0.26 }, { "name": "banana", "color": "yellow", "weight": 0.18 }, {

	"name": "strawberry", "color": "red", "weight": 0.03 }] } }
DM script (basic027)	$start('DataMixer') {"Shop Pricelist": {"MessageId": "$value($strcat($replace(**$localdate**,'T-:',''), $round($random * 10000000000)))", "Name": "$value($lower(//store catalogue/name))", "Version": "V$rpad(//store catalogue/version,'000',5)", "Date": "$value($replace(//store catalogue/date, '-', '/'))", "Number of fruits": $value($count(//name) - $count(//store catalogue/name)), "Total weight of fruit": $value($sum(//weight) } } $endstart
Doeldata	{"Shop Pricelist": {**"MessageId": "20240518110359798502965"**, "Name": "greengrocer supermarket ltd", "Version": "V2.300", "Date": "15/03/2024", "Number of fruits": 3, "Total weight of fruit": 0.47 } }

$isodate

De functie (of beter gezegd globale) $isodate werkt op dezelfde manier als $localdate, waarmee eveneens de huidige datum en tijd (systeemdatum) wordt teruggeven, alleen dan in iso formaat (UTC):

Functie	**$isodate**
Effect	Bepalen van de huidige systeemdatum & tijd
Input	-
Aanroep	$isodate let op, geen haakjes, het is feitelijk een globale variabele
Output	Datum+tijd in formaat YYYY-MM-DDThh:mi:ssZ
Voorbeeld	$isodate levert 18-05-2024T08:49:32Z op

DM Programmeer Script: Programma Structuren

In dit deel worden enkele programmeerstructuren uitgelegd zodat de brondata repeterend (loop) en/of conditioneel (filter, if en case) kan worden doorlopen ter verwerking tot de gewenste doeldata.

We gaan in de volgende voorbeelden uit van een iets uitgebreidere "store catalogue" die hieronder staat, om niet telkens bij ieder voorbeeld opnieuw te vermelden:

```
{"store catalogue":
 {
 "name": "Greengrocer supermarket Ltd",
 "version": 2.3,
 "date": "15-03-2024",
 "fruit":   [
              {
               "name": "apple",
               "color": "red",
               "weight": 0.26,
               "price": {
                "GBP": 0.50
               },
               "imported": false
              },
              {
               "name": "banana",
               "color": "yellow",
               "weight": 0.18,
               "price": {
                "GBP": 0.30
               },
               "imported": true
              },
              {
               "name": "strawberry",
               "color": "red",
               "weight": 0.03,
               "price": {
                "GBP": 0.15
               },
               "imported": false
              }
  ]
 }
}
```

(basic004)

$if

De functie $if is bedoeld om een voorwaardelijk (een deel van) de brondata te verwerken tot de doeldata.

Dit zijn de verplichte en optionele mogelijkheden van de if-functie:

- Verplicht: begint altijd met $if-functie
- Verplicht: eindigt altijd met $endif

Functie	**$if/$endif**
Effect	Voorwaardelijk verwerken van gegevens tot de doeldata
Input	Conditie-expressie
Aanroep	$if(conditie)….$endif
Output	true/false
Voorbeeld	$if($substr_before(//store catalogue/version,'.') = 2) levert true op

Voorbeeld :

We verwerken de brondata alleen wanneer de hoofdversie (deel van 2.3 voor de punt) van het bronbestand 2 is, en anders slaan we de gehele verwerking van de brondata over:

Brondata (basic004)	{"store catalogue": … <zie hierboven> }
DM script (basic028)	$start('DataMixer') **$if**($substr_before(//store catalogue/version,'.') = 2) {"Shop Pricelist": {"MessageId": "$value($strcat($replace($localdate,'T-:',''), $round($random * 10000000000)))", "Name": "$value(//store catalogue/name)", "Version": "V$value(//store catalogue/version)", "Date": "$value($replace(//store catalogue/date, '-', '/'))" } } **$endif** $endstart
Doeldata	{"Shop Pricelist": {"MessageId": "20240518215735136628929 7", "Name": "Greengrocer supermarket Ltd", "Version": "V2.3", "Date": "15/03/2024" } }

$loop

De functie $loop is bedoeld om een repeterende gedeelte uit de brondata te doorlopen om de gegevens te converteren.

Dit zijn de verplichte en optionele mogelijkheden van de loop-functie:

- Verplicht: begint altijd met $loop-functie
- Optioneel (*): $array-functie
- Optioneel: $orderby(**) om de loop in een bepaalde volgorde te doorlopen
- Verplicht: eindigt altijd met $endloop

(*) Alleen verplicht bij **JSON brondata met een array**, bij alle andere brondata zoals XML, CSV mag deze niet aanwezig zijn. Gebruik is dan: $loop(**$array**(folderexpressie))

(**) De $orderby kan/mag ook vaker dan één keer worden gebruikt, b.v. door de volgorde na de 1e $orderby door een 2e $orderby aan te scherpen (omdat 2 sorteercriteria van toepassing zijn).

Functie	**$loop/$endloop**
Effect	Doorlopen/verwerken van een repeterende gedeelte van de brondata
Input	Folderexpressie
Aanroep	$loop(folderexpressie)$orderby(folderexpressie)….$endloop
Output	-
Voorbeeld	$loop($array(//store catalogue/fruit))hi$endloop levert “hihihi” op (voor elk artikel 1x hi)

Algemene syntax $loop in diverse situaties:

Situatie	**Syntax**
Bij XML en CSV brondata	$loop(folderexpressie)… $endloop
Bij JSON brondata	$loop($array(folderexpressie)) … $endloop
Bij gebruik van sortering	$loop(folderexpressie)$orderby(folderexpressie)…$endloop
Bij gebruik van 2 sorteringen	$loop(folderexpressie)$orderby(folderexpressie) $orderby(folderexpressie)…$endloop

Binnen de loop-functie (dus binnen $loop en $endloop) zijn de volgende functies (of beter gezegd: variabelen) mogelijk:

- $index: deze geeft in een loop het iteratie nummer (getal) weer. D.w.z. het nummer voor de hoeveelste keer op dat moment de loop wordt doorlopen), en begint bij 1.
- $last: deze geeft in een loop aan wat het totaal aantal iteraties (getal) gaat woorden. D.w.z. het nummer voor hoeveel keer de loop totaal zal worden doorlopen

Voorbeeld $loop:

Alle fruitartikelen worden doorlopen om de naam en de prijs in EUR te converteren, we beginnen eerst met de naam:

Brondata (basic004)	{"store catalogue": … <zie hierboven> }
DM script (basic029)	$start('DataMixer') {"Shop Pricelist": {"MessageId": "$value($strcat($replace($localdate,'T-:',''), $round($random * 10000000000)))", "Name": "$value(//store catalogue/name)", "Version": "V$value(//store catalogue/version)", "Date": "$value($replace(//store catalogue/date, '-', '/'))", "Articles": [**$loop($array**(//store catalogue/fruit)) {"Name":"$value(name)"},**$endloop**] } } $endstart
Doeldata	{"Shop Pricelist": {"MessageId": "20240518124150631338 9896", "Name": "Greengrocer supermarket Ltd", "Version": "V2.3", "Date": "15/03/2024", "Articles": [**{"Name":"apple"},** **{"Name":"banana"},** **{"Name":"strawberry"},**] } }

Merk op dat in de loop alle 3 fruitartikelen doorloopt, en via de expressie {"Name":"$value(name)"}, netjes elk element begint met "{" en eindigt met "},"

Probleem hierbij is de laatste (rood gemarkeerde) komma (,): deze hoort daar niet te staan. Om het resultaat (zonder laatste komma) te realiseren, kan gebruik worden gemaakt van de functie/variabele $index of $last:

Voorbeeld $index en $last:

Alle fruitartikelen worden doorlopen om de naam en de prijs in EUR te converteren, we beginnen eerst met de naam, waarbij de laatste komma wordt weggelaten via $index en $last: wanneer $index namelijk gelijk is $last (dat is de laatste keer dat de loop wordt doorlopen), moet de komma niet meer worden meegenomen in de doeldata. Dus omgekeerd: de komma moet alleen in de doeldata terechtkomen wanneer $index ongelijk is aan $last (het ongelijkteken is !=)

Brondata (basic004)	{"store catalogue": … <zie hierboven> }
DM script (basic030)	$start('DataMixer') {"Shop Pricelist": {"MessageId": "$value($strcat($replace($localdate,'T-:',''), $round($random * 10000000000)))", "Name": "$value(//store catalogue/name)", "Version": "V$value(//store catalogue/version)", "Date": "$value($replace(//store catalogue/date, '-', '/'))", "Articles": [$loop($array(//store catalogue/fruit)) {"Name":"$value(name)"}$if(**$index**!=**$last**),$endif$endloop] } } $endstart
Doeldata	{"Shop Pricelist": {"MessageId": "20240518222135510046492 9", "Name": "Greengrocer supermarket Ltd", "Version": "V2.3", "Date": "15/03/2024", "Articles": [{"Name":"apple"}, {"Name":"banana"}, {"Name":"strawberry"}] } }

PS. Alternatieve oplossing waarbij alleen $index wordt gebruikt, en $last niet perse nodig is, is door de conditie vooraf in de loop te plaatsen, zodat de komma vooraan (i.p.v. achter) een nieuw element komt te staan, alleen de 1e keer niet:

```
$start('DataMixer')
{"Shop Pricelist":
 {"MessageId": "$value($strcat($replace($localdate,'T-:',''), $round($random *
10000000000)))",
  "Name": "$value(//store catalogue/name)",
  "Version": "V$value(//store catalogue/version)",
  "Date": "$value($replace(//store catalogue/date, '-', '/'))",
  "Articles": [$loop($array(//store catalogue/fruit))
          $if($index!=1),$endif
          {"Name":"$value(name)"}$endloop]
 }
}
$endstart
```

Voorbeeld prijs in €:

Nu moeten we van alle fruitartikelen nog de prijs in EUR verwerken. De prijs wordt in GBP (Great Brittain Pounds) genoemd, de omrekenfactor van GBP naar Euro is 1.16 (1 GPB = 1.16 Euro):

Brondata (basic004)	{"store catalogue": … <zie hierboven> }
DM script (basic031)	$start('DataMixer') {"Shop Pricelist": {"MessageId": "$value($strcat($replace($localdate,'T-:',''), $round($random * 10000000000)))", "Name": "$value(//store catalogue/name)", "Version": "V$value(//store catalogue/version)", "Date": "$value($replace(//store catalogue/date, '-', '/'))", "Articles": [$loop($array(//store catalogue/fruit)) {"Name":"$value(name)", "Price":**{"EUR": $value(1.16 * price/GBP)}**}$if($index!=$last),$endif$endloop] } } $endstart
Doeldata	{"Shop Pricelist": {"MessageId": "20240518223545344419326 3", "Name": "Greengrocer supermarket Ltd", "Version": "V2.3", "Date": "15/03/2024", "Articles": [{"Name":"apple", **"Price":{"EUR": 0.58}**}, {"Name":"banana", **"Price":{"EUR": 0.348}**}, {"Name":"strawberry", **"Price":{"EUR": 0.174}**}] } }

$filter

De $filter functie, die bij de folderexpressies al even is genoemd, kan ook bij de $loop functie optioneel als aanvullende conditie worden gebruikt.

Voorbeeld filter:

We verwerken alleen fruitartikelen die inheems cq. niet geïmporteerd zijn (imported = 'false'). Dit kan uiteraard door in de loop de $if-functie te gebruiken via $if(imported = false) en alleen dan de "Name" en "Price" in de doeldata op te nemen.

Dit kan ook door de conditie in de folderexpressie mee te geven door gebruik te maken van de functie $filter. Alleen de elementen in de loop die voldoen aan de conditie dat het artikel niet is geïmporteerd (imported = false), doen mee in de loop:

Brondata (basic004)	{"store catalogue": … <zie hierboven> }
DM script (basic032)	$start('DataMixer') {"Shop Pricelist": {"MessageId": "$value($strcat($replace($localdate,'T-:',''), $round($random * 10000000000)))", "Name": "$value(//store catalogue/name)", "Version": "V$value(//store catalogue/version)", "Date": "$value($replace(//store catalogue/date, '-', '/'))", "Articles": [$loop($array(//store catalogue/fruit)**$filter(imported='false')**) {"Name":"$value(name)", "Price":{"EUR": $value(1.16 * price/GBP)}}$if($index!=$last),$endif$endloop] } } $endstart
Doeldata	{"Shop Pricelist": {"MessageId": "20240519000714992310 1905", "Name": "Greengrocer supermarket Ltd", "Version": "V2.3", "Date": "15/03/2024", "Articles": [**{"Name":"apple", "Price":{"EUR": 0.58}},** **{"Name":"strawberry", "Price":{"EUR": 0.174}}**] } }

Voorbeeld filter:

We verwerken nu wel alle fruitartikelen, zowel inheemse als geïmporteerd, maar scheiden beide groepen in 2 arrays / 2 nieuwe velden "Imported" en "Native". Dit doen we door de $filter-functie op te nemen in de folderexpressie:

Brondata (basic004)	{"store catalogue": … <zie hierboven> }
DM script (basic033)	$start('DataMixer') {"Shop Pricelist": {"MessageId": "$value($strcat($replace($localdate,'T-:',''), $round($random * 10000000000)))", "Name": "$value(//store catalogue/name)", "Version": "V$value(//store catalogue/version)", "Date": "$value($replace(//store catalogue/date, '-', '/'))", "Articles": {"Native": [$loop($array(//store catalogue/fruit)**$filter(imported='false')**) {"Name":"$value(name)", "Price":{"EUR": $value(1.16 * price/GBP)}}$if($index!=$last),$endif$endloop], "Imported": [$loop($array(//store catalogue/fruit)**$filter(imported='true')**) {"Name":"$value(name)", "Price":{"EUR": $value(1.16 * price/GBP)}}$if($index!=$last),$endif$endloop] } } } $endstart
Doeldata	{"Shop Pricelist": {"MessageId": "20240519001558926914905", "Name": "Greengrocer supermarket Ltd", "Version": "V2.3", "Date": "15/03/2024", "Articles": {**"Native"**: [{"Name":"apple", "Price":{"EUR": 0.58}}, {"Name":"strawberry", "Price":{"EUR": 0.174}}] **"Imported"**: [{"Name":"banana", "Price":{"EUR": 0.348}}], } } }

$groupby

De functie **$groupby** is optioneel onderdeel van de $loop-functie, en kan toegepast worden **in** het **$filter**-statement. Met groupby is het mogelijk om een repeterende gedeelte uit de brondata op een specifiek veld te groeperen om een gewenste structuur in de doeldata te verkrijgen.

In het vorige voorbeeld is enigszins ook een groep-functie gebruikt, om inheems en geïmporteerd fruit uit elkaar te trekken cq. te groeperen. Dit was nog mogelijk via $filter met (vooraf) bekend mogelijke waarden van het veld 'imported', namelijk true of false.

Wanneer een groepering echter op een veld moet plaatsvinden, waarbij de mogelijke waarden (op voorhand) geheel niet bekend zijn, dan biedt de functie '$groupby' hierbij uitkomst.

In onderstaande voorbeeld worden de artikelen gegroepeerd op kleur (veld color). Alle artikelen van dezelfde kleur worden (hiërarchisch) bij elkaar in de doeldata verwerkt. De functie '$groupby' staat binnen in de functie $filter:

Functie	**$groupby**
Effect	Doorlopen/verwerken van een gegroepeerd/repeterend gedeelte van de brondata, de groepering is van toepassing op 1 veld
Input	Folderexpressie
Aanroep	$loop(folderexpressie$filter($groupby(folderexpressie))$orderby(folderexpressie)….$endloop **of** $loop($array(folderexpressie)$filter($groupby(folderexpressie))$orderby(folderexpressie)….$endloop
Output	-
Voorbeeld	$loop($array(//store catalogue/fruit)$filter($groupby(color)))$value(color)$endloop levert "redyellow" op

Het is hierbij ook mogelijk (hoewel dat vrij uitzonderlijk zal zijn) om de groepering te combineren met filtering/condities, zoals in de vorige paragrafen uiteengezet.

Voorbeeld	$loop($array(//store catalogue/fruit)$filter(weight>0.25 $and $groupby(color)))$value(color)$endloop levert alleen "red" op

Hieronder wordt een voorbeeld uitgewerkt waarbij alle kleuren gegroepeerd in de doeldata wordt verwerkt d.m.v. een 2^{e} loop die gebruikt maakt van de (unieke

/ distinct) kleuren in de brondata, waarbij gebruikt wordt gemaakt van een hulp-variabele 'thiscolor' die de (unieke) kleur uit de 1e loop gebruikt in de 2e loop.

Brondata (basic004)	{"store catalogue": … <zie hierboven> }
DM script (basic042)	$start('DataMixer') {"Shop Pricelist": { "MessageId": "$value($strcat($replace($localdate,'T-:',''), $round($random * 10000000000)))", "Name": "$value(//store catalogue/name)", "Version": "V$value(//store catalogue/version)", "Date": "$value($replace(//store catalogue/date, '-', '/'))", "Colors": [**$loop($array(//store catalogue/fruit)$filter($groupby(color)))** $var(thiscolor=$value(color)) {"Color": "$value($thiscolor)", "Articles": [$loop($array(//store catalogue/fruit)$filter(color=**$thiscolor**)) {"Name":"$value(name)", "Price":{"EUR": $value(1.16 * price/GBP)}}$if($index!=$last),$endif$endloop] } $if($index!=$last),$endif$endloop] } } $endstart
Doeldata	{"Shop Pricelist": { "MessageId": "20250112210914473540408 1", "Name": "Greengrocer supermarket Ltd", "Version": "V2.3", "Date": "15/03/2024", "Colors": [{"Color": "red", "Articles": [{"Name":"apple", "Price":{"EUR": 0.58}}, {"Name":"strawberry", "Price":{"EUR": 0.174}}] } , {"Color": "yellow", "Articles": [{"Name":"banana", "Price":{"EUR": 0.348}}] }] } }

$orderby

De functie $orderby is optioneel onderdeel van de $loop-functie, en is bedoeld om een repeterende gedeelte uit de brondata in een gewenste volgorde te doorlopen om de gegevens te converteren.

In de uitleg bij de $loop-functie worden de gegevens in de lus in de volgorde uit de brondata, "van boven naar beneden", doorlopen. Met $orderby is het mogelijk om die volgorde aan te passen:

- Verplicht: begint altijd met $loop-functie
- Optioneel (*): $array-functie
- Optioneel: $orderby(**) om de loop in een bepaalde volgorde te doorlopen
- Verplicht: eindigt altijd met $endloop

(*) Alleen verplicht bij **JSON brondata met een array**, bij alle andere brondata zoals XML, CSV mag deze niet aanwezig zijn. Gebruik is dan: $loop(**$array**(folderexpressie))

(**) De $orderby kan/mag ook vaker dan één keer worden gebruikt, b.v. door de volgorde na de 1e $orderby door een 2e $orderby aan te scherpen (omdat 2 sorteercriteria van toepassing zijn).

Functie	**$orderby**
Effect	Het in bepaalde volgorde doorlopen/verwerken van een repeterende gedeelte van de brondata
Input	1. Folderexpressie 2. DataType (txt|num) 3. Sorting (asc|desc)
Aanroep	$orderby(folderexpressie, datatype, sorting)
Output	-
Voorbeeld	$loop($array(//store catalogue/fruit))$orderby(price, num, asc) doorloopt de fruitartikelen in volgorde van prijs, van laag naar hoog (ascending).

Voorbeeld filter:

We verwerken alle fruitartikelen in twee groepen voor inheemse als geïmporteerd, en verwerken de artikelen in volgorde van prijs, van laag naar hoog:

Brondata (basic004)	{"store catalogue": … <zie hierboven> }

DM script (basic034)	$start('DataMixer') {"Shop Pricelist": {"MessageId": "$value($strcat($replace($localdate,'T-:',''), $round($random * 10000000000)))", "Name": "$value(//store catalogue/name)", "Version": "V$value(//store catalogue/version)", "Date": "$value($replace(//store catalogue/date, '-', '/'))", "Articles": {"Native": [$loop($array(//store catalogue/fruit)$filter(imported='false'))**$orderby(price, num, asc)** {"Name":"$value(name)", "Price":{"EUR": $value(1.16 * price/GBP)}}$if($index!=$last),$endif$endloop], "Imported": [$loop($array(//store catalogue/fruit)$filter(imported='true'))**$orderby(price, num, asc)** {"Name":"$value(name)", "Price":{"EUR": $value(1.16 * price/GBP)}}$if($index!=$last),$endif$endloop] } } } $endstart
Doeldata	{"Shop Pricelist": {"MessageId": "20240519120953610260608 1", "Name": "Greengrocer supermarket Ltd", "Version": "V2.3", "Date": "15/03/2024", "Articles": {"Native": [{"Name":"strawberry", "Price":{"EUR": **0.174**}}, {"Name":"apple", "Price":{"EUR": **0.58**}}], "Imported": [{"Name":"banana", "Price":{"EUR": **0.348**}}] } } }

$case

De functie $case is bedoeld om in geval (meerdere) afhankelijkheden en voorwaarden (condities) de brondata tot de juiste verwerking doeldata te verwerken. Het is mogelijk om in geval van voorwaardelijkheden gebruik te maken van $if, zoals eerder uitgelegd, echter wanneer er niet één maar meerdere voorwaarden in een wat ingewikkeldere structuur van toepassing zijn, dan is het mogelijk om dat eenvoudiger via de $case functie te laten verlopen (i.p.v. veel $if-functies).

Dit zijn de verplichte en optionele mogelijkheden van de case-functie:

- Verplicht: begint altijd met $case-functie
- Verplicht: vervolg direct met minimaal één $when-functie, afgesloten door $endwhen
- Optioneel: eindig met $default-functie, afgesloten door $enddefault
- Verplicht: eindigt altijd met $endcase

Functie	**$case/$endcase**
Effect	Voorwaardelijk verwerken van gegevens tot de doeldata
Input	Folderexpressie
Aanroep	$case$when(conditie)…$endwhen$when(conditie)…$endwhen$endcase
Output	-
Voorbeeld	Uitwerking hieronder

Voorbeeld $case:

We verwerken alle fruitartikelen in twee groepen voor inheemse als geïmporteerd, en verwerken de kleur van de artikelen in een vertaalde 3 letterige kleurcode "Colorcode". Als er geen vertaling kan worden gevonden, dan wordt de onbekende kleurcode 'UNK' (unknown) opgeleverd.

Brondata (basic004)	{"store catalogue": … <zie hierboven> }
DM script (basic035)	$start('DataMixer') {"Shop Pricelist": {"MessageId": "$value($strcat($replace($localdate,'T-:',''), $round($random * 10000000000)))", "Name": "$value(//store catalogue/name)", "Version": "V$value(//store catalogue/version)", "Date": "$value($replace(//store catalogue/date, '-', '/'))", "Articles": {"Native": [$loop($array(//store catalogue/fruit)$filter(imported='false'))$orderby(price, num, asc) {"Name":"$value(name)",

	"Colorcode": **$case** $when (color = 'red') "RED" $endwhen $when (color = 'yellow') "YLW" $endwhen $when (color = 'green') "GRN" $endwhen $when (color = 'blue') "BLU" $endwhen $when (color = 'orange') "ORA" $endwhen $when (color = 'pink') "PNK" $endwhen $default "UNK" $enddefault **$endcase**, "Price":{"EUR": $value(1.16 * price/GBP)}}$if($index!=$last),$endif$endloop], "Imported": [$loop($array(//store catalogue/fruit)$filter(imported='true'))$orderby(price, num, asc) {"Name":"$value(name)", "Colorcode": **$case** $when (color = 'red') "RED" $endwhen $when (color = 'yellow') "YLW" $endwhen $when (color = 'green') "GRN" $endwhen $when (color = 'blue') "BLU" $endwhen $when (color = 'orange') "ORA" $endwhen $when (color = 'pink') "PNK" $endwhen $default "UNK" $enddefault **$endcase**, "Price":{"EUR": $value(1.16 * price/GBP)}}$if($index!=$last),$endif$endloop] } } } $endstart
Doeldata	{"Shop Pricelist": {"MessageId": "20240519124449156408 6107", "Name": "Greengrocer supermarket Ltd", "Version": "V2.3", "Date": "15/03/2024", "Articles": {"Native": [{"Name":"strawberry", **"Colorcode": "RED"** , "Price":{"EUR": 0.174}}, {"Name":"apple", **"Colorcode": "RED"** , "Price":{"EUR": 0.58}}], "Imported": [{"Name":"banana", **"Colorcode": "YLW"** , "Price":{"EUR": 0.348}}] } } }

DM Programmeer Script: eigen functies

In de voorbeelden en uitleg bij de $loop-functie en met name bij de $case-functie worden dezelfde scriptregels vaker herhaald bij de verwerking van de doeldata. Om te voorkomen dat je vaker dezelfde scriptregels moet herhalen, is het mogelijk om dat onder te brengen in een eigen gedefinieerde functie, waardoor het script er overzichtelijker uitziet en beter te onderhouden is.

$function

De functie $function is bedoeld om een eigen functie te definiëren.Dit zijn de verplichte en optionele mogelijkheden van de $function-functie:

- Verplicht: begint altijd met $function
- Optioneel (*): een of meer $parameter-functies
- Verplicht: eindigt altijd met $endfunction

Functie	**$function/$endfunction**
Effect	Definitie van een eigen functie
Input	Unieke (eigen) functienaam
Aanroep	$function(functienaam)$parameter(x) …$endfunction
Output	-
Voorbeeld	Uitwerking hieronder

Binnen $function en $endfunction zijn de volgende specifieke functies mogelijk:

- $parameter. Om de functie met een parameter aan te roepen, is het nodig om de parameternaam bekend te maken. Voor elke parameter is het nodig om binnen de functiedefinitie de parameternaam te registreren. Dat gebeurt met de functie $parameter.

Een eigen gedefinieerde functie wordt buiten de $start en $endstart geplaatst, daarbinnen is niet teogestaan.

Algemene syntax van de aanroep van een eigen functie met naam 'functienaam' is in onderstaande diverse situaties als volgt:

Situatie	**Syntax**
Aanroep zonder parameters	$functienaam()
Aanroep met één parameter met naam 'x' en waarde 'w'	$functienaam(x=w)
Aanroep met twee parameters met naam 'x' en waarde 'w1' en met naam 'y' en waarde 'w2'	$functienaam(x=w1, y=w2) Of $functienaam(y=w2, x=w1)

Voorbeeld 1

De dubbele scriptregels in het voorbeeld van de $case-functie gaan we in een eigen functie onderbrengen met de naam "Vertaalkleur" met één parameter die we 'x' noemen, zodat we dubbele scriptregels kunnen voorkomen. De doeldata blijft precies hetzelfde, daar verandert niets aan.

DM-script met eigen gedefinieerde functie 'TranslateColor':

```
$start('DataMixer')
{"Shop Pricelist":
 {"MessageId": "$value($strcat($replace($localdate,'T-:',''), $round($random * 10000000000)))",
  "Name": "$value(//store catalogue/name)",
  "Version": "V$value(//store catalogue/version)",
  "Date": "$value($replace(//store catalogue/date, '-', '/'))",
  "Articles": {"Native": [$loop($array(//store catalogue/fruit)$filter(imported='false'))$orderby(price, num, asc)
            {"Name":"$value(name)",
             "Colorcode": $TranslateColor(x=color),
            "Price":{"EUR": $value(1.16 * price/GBP)}}$if($index!=$last),$endif$endloop],
            "Imported": [$loop($array(//store catalogue/fruit)$filter(imported='true'))$orderby(price, num, asc)
            {"Name":"$value(name)",
             "Colorcode": $TranslateColor(x=color),
             "Price":{"EUR": $value(1.16 * price/GBP)}}$if($index!=$last),$endif$endloop]
           }
 }
}
$endstart

$function(TranslateColor)
$/* Color translation into 3 char code $*/
$parameter(x)
$case
  $when ($x = 'red') "RED" $endwhen
  $when ($x = 'yellow') "YLW" $endwhen
  $when ($x = 'green') "GRN" $endwhen
  $when ($x = 'blue') "BLU" $endwhen
  $when ($x = 'orange') "ORA" $endwhen
  $when ($x = 'pink') "PNK" $endwhen
  $default "UNK" $enddefault
$endcase
$endfunction
```

(basic036)

Voorbeeld 2

Het script kan nog verder worden ontdubbeld, waarbij de doeldata verder onveranderd blijft:

1. Statement "$if($index!=$last),$endif" kan compacter, en waardoor het leesbaarder wordt, en vooral omdat dit statement veel vaker in het DM script zal voorkomen, kan het wenselijk zijn om dit 1x te definiëren. Dit doen we via functie 'OptionalComma'.
2. De drie gegevens Name, Colorcode en Price kunnen ook in 1 functie worden gedefinieerd, dat gebeurt via functie 'ArticleData':

```
$start('DataMixer')
{"Shop Pricelist":
 {"MessageId": "$value($strcat($replace($localdate,'T-:',''), $round($random * 10000000000)))",
  "Name": "$value(//store catalogue/name)",
  "Version": "V$value(//store catalogue/version)",
  "Date": "$value($replace(//store catalogue/date, '-', '/'))",
  "Articles": {"Native": [$loop($array(//store catalogue/fruit)$filter(imported='false'))$orderby(price, num, asc)
                $ArticleData()
                $endloop],
          "Imported": [$loop($array(//store catalogue/fruit)$filter(imported='true'))$orderby(price, num, asc)
                $ArticleData()
                $endloop]
          }
 }
}
$endstart

$function(TranslateColor)
$/* Color translation into 3 char code $*/
$parameter(x)
$case
  $when ($x = 'red') "RED" $endwhen
  $when ($x = 'yellow') "YLW" $endwhen
  $when ($x = 'green') "GRN" $endwhen
  $when ($x = 'blue') "BLU" $endwhen
  $when ($x = 'orange') "ORA" $endwhen
  $when ($x = 'pink') "PNK" $endwhen
  $default "UNK" $enddefault
$endcase
$endfunction

$function(OptionalComma)
$if($index!=$last),$endif
$endfunction

$function(ArticleData)
{"Name":"$value(name)",
 "Colorcode": $TranslateColor(x=color),
 "Price":{"EUR": $value(1.16 * price/GBP)}}$OptionalComma()
$endfunction
```

(basic037)

$var

De functie $var is bedoeld om een of meer constanten/variabelen te definiëren, die vaker in het script gebruikt kunnen worden:

- Verplicht: begint altijd met $var
- Minimaal één en maximaal 5 variabele-expressies, gescheiden door komma
- Een variabele expressie bestaat uit:
    - Variabele naam
    - = teken
    - Waarde die wordt toegekend aan de variabele.

Functie	**$var**
Effect	Definitie van een eigen variabele
Input	Variabele naam en waarde
Aanroep	$var(naam=waarde)
Output	-
Voorbeeld	$var(x=3) zet de variabele naam x op waarde 3, en kan worden gebruikt via $x

Een variabele heeft een scope, en is alleen geldig in de functie waarin deze wordt gedefinieerd.

Wanneer een variabele 'x' wordt gedefinieerd tussen $start en $endstart, dan is deze variabele 'x' alleen daar te gebruiken (via aanroep van $x). Deze variabele 'x' kan b.v. niet in een eigen functie wordt gebruikt.

En omgekeerd kan een variabele 'y' die in een eigen functie wordt gedefinieerd, niet in het $start-gedeelte worden gebruikt, en ook niet in andere functies. Dus alleen in de functie waarin deze is gedefinieerd.

Het is wel mogelijk om globale variabelen te definiëren, die wel overal in het DM script (dus in $start en in alle eigen functies) kunnen worden gebruikt. Een globale variabele wordt buiten $start/$endstart en buiten eigen functies gedefinieerd.

Beperking/let op: een variabele in DM script kan maximaal 1x een toekenning van een waarde krijgen, nooit vaker, want dat geeft een foutmelding.

Voorbeeld

De omrekenkoers (factor 1.16 van GBP naar EUR) definiëren we 1x in het script als globale variabele 'exchange_rate'. Datzelfde doen we voor de waarde 10000000000 die wordt gebruikt bij de random functie, en zetten die waarde in de variabele 'multiplier':

```
$start('DataMixer')
{"Shop Pricelist":
 {"MessageId": "$value($strcat($replace($localdate,'T-:',''), $round($random * $multiplier)))",
  "Name": "$value(//store catalogue/name)",
  "Version": "V$value(//store catalogue/version)",
  "Date": "$value($replace(//store catalogue/date, '-', '/'))",
  "Articles": {"Native": [$loop($array(//store catalogue/fruit)$filter(imported='false'))$orderby(price, num, asc)
                $ArticleData()
                $endloop],
           "Imported": [$loop($array(//store catalogue/fruit)$filter(imported='true'))$orderby(price, num, asc)
                $ArticleData()
                $endloop]
          }
 }
}
$endstart

$var(exchange_rate = 1.16, multiplier = 10000000000)

$function(TranslateColor)
$/* Color translation into 3 char code $*/
$parameter(x)
$case
  $when ($x = 'red') "RED" $endwhen
  $when ($x = 'yellow') "YLW" $endwhen
  $when ($x = 'green') "GRN" $endwhen
  $when ($x = 'blue') "BLU" $endwhen
  $when ($x = 'orange') "ORA" $endwhen
  $when ($x = 'pink') "PNK" $endwhen
  $default "UNK" $enddefault
$endcase
$endfunction

$function(OptionalComma)
$if($index!=$last),$endif
$endfunction

$function(ArticleData)
{"Name":"$value(name)",
 "Colorcode": $TranslateColor(x=color),
 "Price":{"EUR": $value($exchange_rate * price/GBP)}}$OptionalComma()
$endfunction
```

DM Programmeer Script: logische functies

In een conditionele expressie, die b.v. wordt in de $if-functie of $filter-functie, kunnen de logische operanden $and, $or en $not wordten toegepast.

$and

De functie $and is bedoeld om een voorwaardelijk (een deel van) de brondata te verwerken tot de doeldata

Functie	**$and**
Effect	Voorwaardelijk verwerken van gegevens tot de doeldata
Input	Conditie-expressie
Aanroep	conditie1 $and conditie2
Output	true/false
Voorbeeld	$value($substr_before(//store catalogue/version,'.') = 2 $and $substr_after(//store catalogue/version,'.') = 3) levert true op

$or

De functie $or is bedoeld om een voorwaardelijk (een deel van) de brondata te verwerken tot de doeldata

Functie	**$or**
Effect	Voorwaardelijk verwerken van gegevens tot de doeldata
Input	Conditie-expressie
Aanroep	conditie1 $or conditie2
Output	true/false
Voorbeeld	$value($substr_before(//store catalogue/version,'.') = 2 $or $substr_after(//store catalogue/version,'.') = 4) levert true op

$not

De functie $not is bedoeld om een voorwaardelijk (een deel van) de brondata te verwerken tot de doeldata

Functie	**$or**
Effect	Voorwaardelijk verwerken van gegevens tot de doeldata
Input	Conditie-expressie
Aanroep	$not(conditie1)
Output	true/false
Voorbeeld	$value($not($substr_before(//store catalogue/version,'.') = 2)) levert false

Voorbeeld :

We verwerken alleen de 'goedkopere' artikelen met een prijs t/m GBP. 0,40. De artikelen die een hogere prijs hebben, willen we niet in de doeldata verwerken. Verder vervangen we het teken '!=' in de functie OptionalComma door het teken '=', i.c.m. de functie $not:

```
$start('DataMixer')
{"Shop Pricelist":
 {"MessageId": "$value($strcat($replace($localdate,'T-:',''), $round($random * $multiplier)))",
  "Name": "$value(//store catalogue/name)",
  "Version": "V$value(//store catalogue/version)",
  "Date": "$value($replace(//store catalogue/date, '-', '/'))",
  "Articles": {"Native": [$loop($array(//store catalogue/fruit)$filter(imported='false' $and price/GBP <=
0.40))$orderby(price, num, asc)
                $ArticleData()
                $endloop],
          "Imported": [$loop($array(//store catalogue/fruit)$filter(imported='true' $and price/GBP <=
0.40))$orderby(price, num, asc)
                $ArticleData()
                $endloop]
          }
 }
}
$endstart
$var(exchange_rate = 1.16, multiplier = 10000000000)

$function(TranslateColor)
$/* Color translation into 3 char code $*/
.....
.....
$endfunction

$function(OptionalComma)
$if($not($index=$last)),$endif
$endfunction

$function(ArticleData)
{"Name":"$value(name)",
 "Colorcode": $TranslateColor(x=color),
 "Price":{"EUR": $value($exchange_rate * price/GBP)}}$OptionalComma()
$endfunction
```

(basic039)

In de doeldata ontbreekt artikel 'appel', want deze is met 0,50 GBP duurder dan de 'toegestane' 0,40 GBP:


```
{"Shop Pricelist":
 {"MessageId": "20240521233018812106933 7",
  "Name": "Greengrocer supermarket Ltd",
  "Version": "V2.3",
  "Date": "15/03/2024",
  "Articles": {"Native": [
{"Name":"strawberry",
 "Colorcode":  "RED" ,
 "Price":{"EUR": 0.174}}],
            "Imported": [
{"Name":"banana",
 "Colorcode":  "YLW" ,
 "Price":{"EUR": 0.348}}]
          }
 }
}
```

DM Programmeer Script: speciale functies

$messageid

Met de functie (of beter gezegd globale) $messageid wordt een unieke cijferreeks verkregen:

Functie	**$messageid**
Effect	Bepalen van een unieke cijferreeks
Input	-
Aanroep	$messageid let op, geen haakjes, het is feitelijk een globale variabele
Output	nnnnnnnnnnnnnnnnnnnnnnnnnnnnnn (30+ cijfers)
Voorbeeld	$messageid levert b.v. 74128352202405221944243656752313 op

Voorbeeld :

We gaan het veld "MessageId" aanpassen door gebruik te maken van $messageid, zodat elke te versturen bericht een unieke ID meekrijgt:

Brondata
(basic003)
	{"store catalogue":
	{
	"name": "Greengrocer supermarket Ltd",
	"version": 2.3,
	"date": "15-03-2024",
	"fruit":
	[
	{
	"name": "apple",
	"color": "red",
	"weight": 0.26
	},
	{
	"name": "banana",
	"color": "yellow",
	"weight": 0.18
	},
	{
	"name": "strawberry",
	"color": "red",
	"weight": 0.03
	}
	]
	}
	}

DM script (basic040)	$start('DataMixer') {"Shop Pricelist": {"MessageId": "$value(**$messageid**)", "Name": "$value($lower(//store catalogue/name))", "Version": "V$rpad(//store catalogue/version,'000',5)", "Date": "$value($replace(//store catalogue/date, '-', '/'))", "Number of fruits": $value($count(//name) - $count(//store catalogue/name)), "Total weight of fruit": $value($sum(//weight) } } $endstart
Doeldata	{"Shop Pricelist": {"MessageId": "**74128352202405221944243656752313**", "Name": "greengrocer supermarket ltd", "Version": "V2.300", "Date": "15/03/2024", "Number of fruits": 3, "Total weight of fruit": 0.47 } }

Wanneer in de DM de functie $messageid wordt gebruikt, dan levert dat telkens een andere cijferreeks op, zodat elk gebruik ervan tot een unieke reeks zal leiden.

Speciale tekens: $cr, $lf, $sp, $apos

Het kan in speciale gevallen voorkomen dat een carriage return ($cr), line feed ($lf), spatie ($sp) of apostrof ($apos) niet goed of helemaal niet in de doeldata terecht kan komen. Dat is m.n. het geval wanneer vaste (constante) tekst in het DM script staat, in b.v. de functie $strcat om teksten te concateneren.

In die gevallen is het mogelijk om via de overeenkomstige functie het betreffende teken te forceren:

Teken	DM functie
Carriage Return	$cr
Line feed	$lf
Spatie	$sp
Apostrof	$apos

Voorbeeld

Het concateneren van verschillende strings (via $strcat) met speciale tekens kan het beste plaatsvinden m.b.v. bovenstaande functies $cr, $lf, $sp en $apos.

Het resultaat zonder gebruik van de speciale teken-functies zal in de volgende notatie een error geven, waardoor er geen doeldata wordt verkregen:

```
$value($strcat(//store catalogue/name, ' is a store in London city.','You're welcome, see you!'))
```

Probleem zit in de (rood gemarkeerde) apostrof in You'er, deze moet in een vaste tekst vervangen worden door $apos en tevens (als uitzondering) worden losgekoppeld van de tekst en als aparte parameter op te voeren, waarmee de notatie dan wordt:

```
$value($strcat(//store catalogue/name, ' is a store in London city.','You', $apos, 're welcome, see you!'))
```

Echter ook deze notatie levert deze output op, en is ook niet het gewenste resultaat:

```
Greengrocer supermarket LtdisastoreinLondoncity.You'rewelcome,seeyou!
```

Alle spaties in de vaste teksten zijn namelijk verdwenen, en moeten door $sp worden vervangen, waarmee de notatie dan wordt:

```
$value($strcat(//store catalogue/name,
'$spis$spa$spstore$spin$spLondon$spcity.', '$spYou', $apos,
're$spwelcome,$spsee$spyou!'))
```

Of wellicht iets leesbaarder:

```
$value($strcat(//store catalogue/name, ' $sp is $sp a $sp store $sp in
$sp London $sp city.', ' $sp You', $apos, 're $sp welcome, $sp see $sp
you!'))
```

Hiermee wordt wel de gewenste doeldata verkregen:

```
Greengrocer supermarket Ltd is a store in London city. You're welcome,
see you!
```

$coalesce

De functie $coalesce is een erg eenvoudige notatie, bedoeld om een voorwaardelijk (een deel van) de brondata te verwerken tot de doeldata.

$coalesce kent 5 parameters, en levert de eerste parameter als waarde op die niet leeg is. Hiermee wordt in principe op eenvoudige wijze, in 1 regel, de volgende $case constructie nagebootst:

```
$case
$when (parameter1 != '') $value(parameter1) $endwhen
$when (parameter2 != '') $value(parameter2) $endwhen
$when (parameter3 != '') $value(parameter3) $endwhen
$when (parameter4 != '') $value(parameter4) $endwhen
$when (parameter5 != '') $value(parameter5) $endwhen
$endcase
```

Dit zijn de verplichte en optionele mogelijkheden van de $coalesce-functie:

- Verplicht: begint altijd met $coalesce-functie
- Verplicht: heeft minimaal 1 parameter
- Optioneel: heeft een 2e parameter, evt. een 3e parameter, evt. een 4e en evt. een 5e parameter
- Verplicht: heeft maximaal 5 parameters

Functie	**$coalesce**
Effect	Voorwaardelijk verwerken van gegevens tot de doeldata
Input	Maximaal 5 folderexpressies
Aanroep	$coalesce(folder1, folder2, folder3, folder4, folder5)
Output	Waarde van folder1 indien deze niet leeg is, en anders waarde van folder 2 indien deze niet leeg is, etc
Voorbeeld	Zie uitwerking hieronder

Voorbeeld :

De versie van de store catalogus staan in veld 'version', echter in de toekomst komt de versie in het veld 'release' te staan. T.b.v. de compatibiliteit, om de DM zowel in de huidige als ook alvast in de toekomstige brondata te laten werken, passen we de $coalesce functie toe:

Brondata nu:

Brondata nu
(basic001)
	{"store catalogue":
	{
	"version": 2.3,
	"date": "15-03-2024"
	}

DM script (basic041)	$start('DataMixer') {"Shop Pricelist": {"MessageId": "$value($messageid)", "Version": "V**$coalesce**(//store catalogue/release,//store catalogue/version)" } } $endstart
Doeldata	{"Shop Pricelist": {"MessageId": "4646601520240522205959539642914 4", "Version": "**V2.3**" } }

Brondata toekomstig:

Brondata nu
(basic005)
	{"store catalogue":
	{
	"release": 3.1,
	"date": "15-06-2024"
	}

DM script (basic041)	$start('DataMixer') {"Shop Pricelist": {"MessageId": "$value($messageid)", "Version": "V**$coalesce**(//store catalogue/release,//store catalogue/version)" } } $endstart
Doeldata	{"Shop Pricelist": {"MessageId": "33945932024052221053534475 7436", "Version": "**V3.1**" } }

$lpad

De functie $lpad wordt gebruikt om een string aan de linkerzijde ervan aan te vullen met andere tekens, tot een maximale lengte is bereikt:

Functie	**$lpad**
Effect	Een tekst aan de linkerzijde aanvullen met een (of meer) character(s)/ teken(s).
Input	Brontekst (string), Aanvultekst (string), maximale lengte (integer)
Aanroep	$lpad (folderexpressie, aanvultekst, lengte)
Output	Aangevulde tekst
Voorbeeld	$lpad(//store catalogue/version, '000',5) levert 002.3 op

Voorbeeld :

Het versienummer moet bestaan uit maximaal 5 tekens, en wordt voorafgegaan door eventuele nullen:

Brondata (basic002)	{"store catalogue":
	{
	"name": "Greengrocer supermarket Ltd",
	"version": 2.3,
	"date": "15-03-2024"
	}
	}

DM script (basic015)	$start('DataMixer') {"Shop Pricelist": {"Name": "$value($lower(//store catalogue/name))", "Version": "V**$lpad**(//store catalogue/version,'000',5)", "Date": "$value($replace(//store catalogue/date, '-', '/'))" } } $endstart
Doeldata	{"Shop Pricelist": {"Name": "greengrocer supermarket ltd", **"Version": "V002.3"**, "Date": "15/03/2024" } }

Bijzonderheden:

De functie $lpad kan als één van de weinige functies **niet** wordt voorafgegaan door de functie $value (dat levert namelijk een foutmelding op).

Wanneer de lengte van de brontekst groter of gelijk is aan parameter "lengte", dan wordt de brontekst zelf ingekort tot het maximum aantal tekens.

$rpad

De functie $rpad wordt gebruikt om een string aan de linkerzijde ervan aan te vullen met andere tekens, tot een maximale lengte is bereikt:

Functie	**$rpad**
Effect	Een tekst aan de rechterzijde aanvullen met een (of meer) character(s)/ teken(s).
Input	Brontekst (string), Aanvultekst (string), maximale lengte (integer)
Aanroep	$rpad (folderexpressie, aanvultekst, lengte)
Output	Aangevulde tekst
Voorbeeld	$rpad(//store catalogue/version, '000',5) levert 2.300 op

Voorbeeld :

Het versienummer moet bestaan uit maximaal 5 tekens, en wordt beëindigd met eventuele nullen:

Brondata (basic002)	{"store catalogue":
	{
	"name": "Greengrocer supermarket Ltd",
	"version": 2.3,
	"date": "15-03-2024"
	}
	}

DM script (basic016)	$start('DataMixer') {"Shop Pricelist": {"Name": "$value($lower(//store catalogue/name))", "Version": "V**$rpad**(//store catalogue/version,'000',5)", "Date": "$value($replace(//store catalogue/date, '-', '/'))" } } $endstart
Doeldata	{"Shop Pricelist": {"Name": "greengrocer supermarket ltd", **"Version": "V2.300",** "Date": "15/03/2024" } }

Bijzonderheden:

De functie $lpad kan als een van de weinige functies **niet** wordt voorafgegaan door de functie $value (dat levert namelijk een foutmelding op).

Wanneer de lengte van de brontekst groter of gelijk is aan parameter "lengte", dan wordt de brontekst zelf ingekort tot het maximum aantal tekens.

$namespace

Met de functie $namespace kan worden gebruikt in de conversie waarin een XML formaat is betrokken. Dat kan zijn in geval van een XML als brondata of in geval van een XML als doeldata.

Dit zijn de verplichte en optionele mogelijkheden van de namespace-functie:

- Verplicht: begint altijd met $namespace-functie
- Verplicht: vervolg direct met minimaal één namespace expressie
- Verplicht: eindigt altijd met $endnamespace

Functie	**$namespace/$endnamespace**
Effect	Het genereren van een named of unnamed namespace in de XML als doeldata, of voor het kunnen uitlezen van een XML als brondata die namespaces bevat
Input	Namespace expressie
Aanroep	$namespaces xmlns="http://www.grocerycatalogueformat.com" $endnamespaces
Output	
Voorbeeld	Zie casussen XML-XML

Mixen van meerdere databronnen

In de sectie wordt beschreven hoe meerdere databronnen met elkaar kunnen worden gemixt (mergen) en naar de gewenste doeldata kan worden geschreven.

Eigen databronnen

Het is hierbij toegestaan om verschillende typen databronnen met elkaar te mixen. Oftewel het is mogelijk om een XML, een Json en een CSV met elkaar te mixen, om daarna via een DM$cript de doeldata te genereren.

In paragraaf Aanroep DM service is de aanroep van DM webservice uitgelegd. In de array "data" kunnen meerdere (verschillende) bronbestanden worden meegegeven. Hieronder een voorbeeld van 3 verschillende brondata typen (Json, Json en XML). Het is daarbij (optioneel) mogelijk om een "name" mee te kunnen geven. Dat kan praktisch zijn (of zelfs noodzakelijk) als dezelfde foldernaam in de brondata kan voorkomen. In dat geval is het mogelijk om met de "name" het onderscheid duidelijk te maken.

Hieronder een voorbeeld van 3 bestanden, naast de bekende 'basic004.json'…

```
{"store catalogue":
 {
 "name": "Greengrocer supermarket Ltd",
 "version": 2.3,
 "date": "15-03-2024",
 "fruit":   [
              {
               "name": "apple",
               "color": "red",
```

… zijn er 2 bestanden (een Json en een XML) die tezamen enkele kleurcodes bevatten:

```
{"colorcodes": [{"color": "red", "code": "red"}, {"color": "green", "code": "grn"}]}
```

```
<colorcodes>
<color desc="yellow">ylw</color>
<color desc="pink">pnk</color>
</colorcodes>
```

Deze 3 bestanden kan als volgt via DataMixer worden gemixt:

```
{
        "DM": {
                "data": [
                        {
                                "name": "main",
                                "content": "hier staat uw brondata 1 in b.v. xml"
                        },
                        {
                                "name": "colortable_1",
                                "content": "hier staat uw brondata 2 in b.v. json"
                        },
                        {
                                "name": "colortable_2",
                                "content": "hier staat uw brondata 3 in b.v. xml"
                        }
                ],
                "dm$": "… hier staat DM $script om uw brondata te converteren …  "
        }
}
```

In dit voorbeeld zijn de aantal kleurcodes (uiteraard) gering gehouden, het gaat hier om het principe. Doordat het mogelijk is om deze bestanden deel te laten maken in DataMixer, kan b.v. de volgende functie TranslateColor…

```
$/* Color translation into 3 char code $*/
$function(TranslateColor)
$parameter(x)
 $case
   $when ($x = 'red')RED$endwhen
   $when ($x = 'yellow')YLW$endwhen
   $when ($x = 'green')GRN$endwhen
   $when ($x = 'blue')BLU$endwhen
   $when ($x = 'orange')ORA$endwhen
   $when ($x = 'pink')PNK$endwhen
   $defaultUNK$enddefault
 $endcase
$endfunction
```

… uit bovenstaande scripts worden vervangen door de volgende versie:

```
$/* Color translation into 3 char code $*/
$function(TranslateColor)
$var(c1=
$value(//colortable_1/$array(colorcodes)$filter(color=$x)/code))
$var(c2=
$value(//colortable_2/colorcodes/color$filter($getattr(desc)=$x)))
$if($c1!='')$value($c1)$endif
$if($c2!='')$value($c2)$endif
$endfunction
```

… waardoor de harde codering in DM$cript van de kleurcode-vertaling kan worden vervangen door de waarden die door de brondata (Json en XML) wordt aangeleverd.

Standaard databronnen

Bij de aanroep van DM webserver is het verder ook nog mogelijk om naast uw eigen databron of een mix van databronnen ook standaard databronnen te 'includen'. Deze standaard databronnen kunnen automatisch worden betrokken via de functies $iref en $xref in het DM$script.

$iref

Met de functie $iref wordt een standaard databron van DataMixer (d.w.z. een interne databron die beschikbaar wordt gesteld door DataMixer) betrokken / geïmporteerd zodat het DM$cript hier handig gebruik van kan maken:

Functie	**$iref**
Effect	Includen/importeren van een interne (standaard) databron
Input	iref-code Mogelijke waarden voor iref-code zijn o.a.: colortable postalcodesnl periodieksysteem PS. De actuele lijst met iref-codes is te vinden op de website van DataMixer
Aanroep	$iref(iref-code) let op, aanroep met een bekende iref-code resulteert in includen/importeren van de bijbehorende standaard databron, een onbekende iref-code heeft geen effect.
Output	Interne (standaard) databron wordt ge-include, waar het DM$cript gebruik van kan maken
Voorbeeld	$iref(colortable) include/importeert het interne/standaard kleurenbestand van DataMixer. Aanroep van b.v. $value(//colortable/csv$filter(Color='Tomato')/Hex) levert #ff6347 op

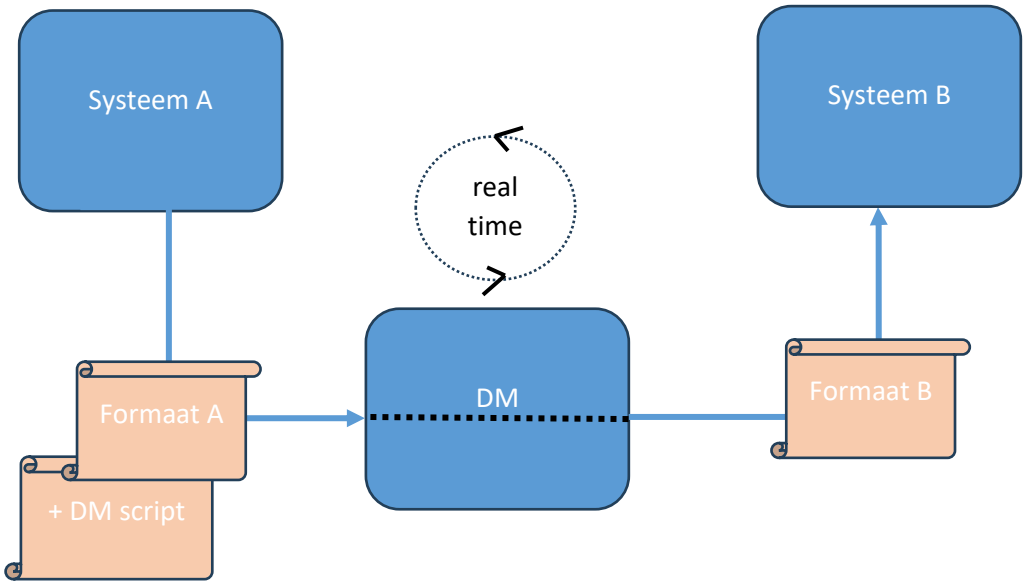
$xref

Met de functie $xref wordt een standaard databron buiten DataMixer (d.w.z. een externe /publieke databron die beschikbaar wordt gesteld door een derde partij) betrokken / geïmporteerd zodat het DM$cript hier handig gebruik van kan maken:

Functie	**$xref**
Effect	Includen/importeren van een externe (standaard) databron
Input	xref-code Mogelijke waarden voor xref-code zijn o.a.: eur2usd eur2jpy PS. De actuele lijst met xref-codes is te vinden op de website van DataMixer
Aanroep	$xref(xref-code) let op, aanroep met een bekende xref-code resulteert in includen/importeren van de bijbehorende standaard databron, een onbekende xref-code heeft geen effect.
Output	Externe (standaard) databron wordt ge-include, waar het DM$cript gebruik van kan maken
Voorbeeld	$xref(eur2usd) include/importeert het externe/standaard met actuele valutakoers EUR (Euro) naar USD (US Dollar). Aanroep van b.v. $value(//eur2usd//cb:value) levert de actuele wisselkoers van EUR naar USD van 1.0377 op

Simple MessageBroker

Het is mogelijk om vanuit het bronsysteem (systeem A) de webservice van DM met de brondata (formaat A) en het DM script aan te roepen. Het resultaat, oftewel de doeldata (formaat B) wordt niet als reponse naar systeem A teruggestuurd, maar wordt direct doorgestuurd naar systeem B, waarbij het antwoord (response) van systeem B via DM naar systeem A wordt teruggekoppeld. DM fungeert in deze situatie als een (eenvoudige) messagebroker, waardoor een koppeling van systeem A naar systeem B wordt gelegd.



In paragraaf Aanroep DM service is de aanroep van DM webservice uitgelegd. Om de MessageBroker functie te gebruiken is het nodig om daarvoor extra parameters in de JSON aanroep mee te sturen. De MessageBroker (afgekort mb) heeft een eigen sectie in de JSON (met de overeenkomstige naam 'mb'), met daarin een aantal gegevens die meegestuurd moeten/kunnen worden:

```
{
"DM": {
   "data": [….],
   "dm$": "… hier staat DM $script om uw brondata te converteren …  ",
   "mb": { "url": "https://systeemB.nl/...",
         "method": "POST",
         "contenttype" : "text/xml",
         "soapaction" : "voorbeeldje",
         "authorization": "Basic abcd1234efgh5678",
         "customheader": "X-myheader-name",
         "customhadervalue": "Value-of-customheader",
         "timeout": 60,
         "token": {  "url": "https://identityserver.com/oauth2/..",
                  "contenttype" : "text/plain",
                  "prefix_authorization" : "Bearer",
                  "client_id" : "1234-5678-9012-3456",
                  "client-secret": "9876zyxw",
                  "granttype": "client_credentials",
                  "resource" : "",
                  "scope": "https://systeemB.nl/..."
                }
       }
   }
}
```

De gegevens url, method en contenttype zijn minimaal verplicht, om het resultaat van DataMixer door te kunnen sturen naar een ander endpoint.

Hieronder worden 2 voorbeeld gegeven:

1. Eerst een voorbeeld van een DM$cript, wat als resultaat een DM$script oplevert, en terugstuurt naar de aanroeper,
2. Daarna hetzelfde voorbeeld, maar waarbij het resultaat niet wordt teruggestuurd, maar via de MessageBroker functie wordt doorgestuurd naar het endpoint van nogmaals DataMixer. Kortom: DataMixer wordt in de MessageBroker nogmaals (voor de 2e keer) aangeroepen, waarbij dat resultaat wordt teruggestuurd naar de aanroeper.

Voorbeeld 1:

```
{
"DM": {
   "data": [{"content":"x"}],
   "dm$": "$start('DataMixer'){\"dm\": {\"data\": [{\"content\": \" \"}],\"dm$\": \"$start('DataMixer')Hello World$value($strcat('$','endstart'))\"}}$endstart"
      }
}
```

Het resultaat wat als reponse wordt teruggestuurd is deze JSON:

```
"dm": {
   "data": [{"content": " "}],
   "dm$": "$start('DataMixer')Hello World$endstart"}
}
```

Voorbeeld 2: gebruik van MessageBroker

```
{
"DM": {
   "data": [{"content":"x"}],
   "dm$": "$start('DataMixer'){\"dm\": {\"data\": [{\"content\": \" \"}],\"dm$\":
\"$start('DataMixer')Hello World$value($strcat('$','endstart'))\"}}$endstart",
   "mb": {url: "https://datamixer.nl/api/1.0/?APIKEY=freeversion", "method":
"POST", "contenttype": "text/plain"}
        }
}
```

Het resultaat wat als reponse wordt teruggestuurd is deze tekst:

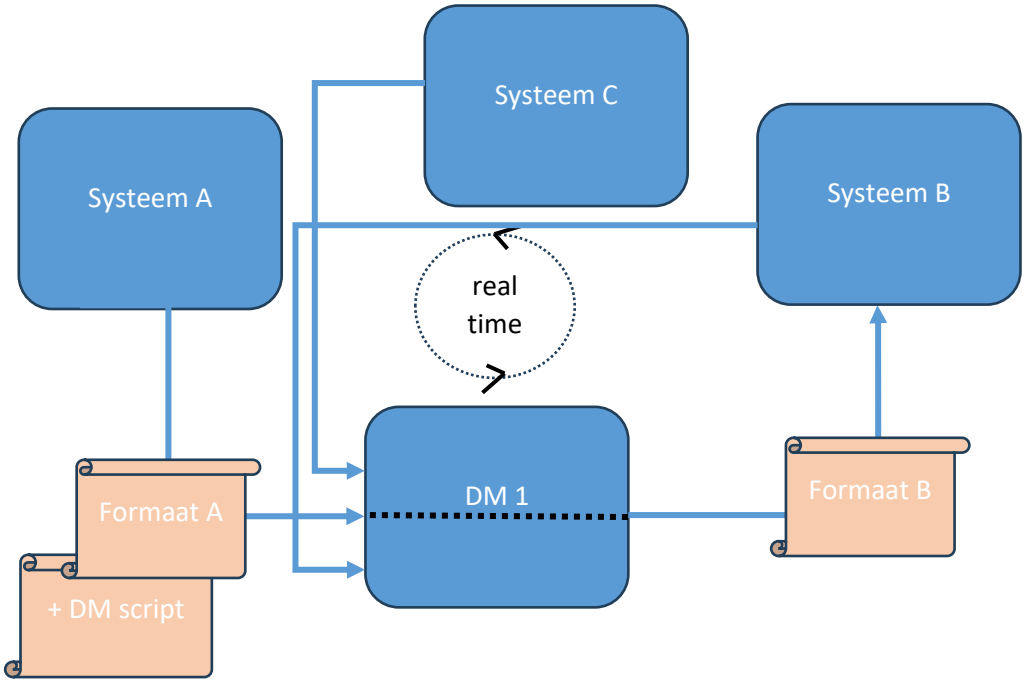
```
Hello World
```

Advanced MessageBroker

Bovenop de "simple" MessageBroker is het mogelijk om vanuit bronsysteem A niet één, maar meerdere sequentiële DM$cripts (achter elkaar) te laten runnen, waarbij ook meerdere systemen C, D, etc (webservers/endpoints) zijn betrokken, waarbij het eindresultaat uiteindelijk naar het doelsysteem B wordt verstuurd. Ook systeem B kan uiteraard (net als C,D etc) ook worden betrokken in de sequentiële aanroepen.

Typische toepassing hierin is dat het formaat B pas naar systeem B kan worden gestuurd, als een waarde in formaat A wordt aangeboden als een interne ID (key of sleutel) die in systeem B bekend is. Of in geval van een vertaling van een waarde die staat in formaat A naar een andere waarde volgens een vertaaltabel in een derde systeem C. In onderstaande figuur zien we 3 systemen:

1. Bronsysteem A om formaat A om te zetten in uiteindelijk doelformaat B om naar systeem B te versturen.
2. In formaat A staat echter een waarde/code 'abc', die in zowel systeem A als systeem B bekend is, maar deze code kan alleen aan systeem B worden aangeboden via de overeenkomstige interne ID/sleutel 123, welke via een REST-API uit systeem B kan worden opgehaald (via een GET)
3. In formaat B moet een waarde uit een andere (3e) systeem C worden toegevoegd.



Dit kan allemaal realtime met 1 aanroep naar DataMixer plaatsvinden, door achtereenvolgens in de aanroep naar de DM service een array van Simple MessageService principe toe te passen.

Hieronder een concreet voorbeeld met een systeem C erbij

1. We vragen eerst uit systeem C (via een GET) de inhoud van een aanvullende gegevens op, en converteren dit tot formaat C (via een bijbehorende DM script. Dit levert dus formaat C, die we hieronder als brondata weer toevoegen.
2. Behalve formaat C (verkregen uit een 3e systeem C, dat had ook systeem B kunnen zijn uiteraard), voegen we ook (bron)formaat A toe en tesamen (formaat A en formaat C) converteren we via een 2e DM script tot formaat B en sturen dat naar systeem B.

Stap 1: ophalen via **GET** van JSON formaat C uit systeem C (= Postman-echo.com), we roepen daartoe DataMixer aan met deze JSON:

```
{
"DM":
   [{mb: {url: "https://postman-echo.com/get?foo1=hallo&foo2=doei", "method": "GET", "contenttype": "text/plain"}}
}
```

Het resultaat wat als response wordt teruggestuurd uit systeem C is deze JSON:


```
{
"args": {
"foo1": "hallo",
"foo2": "doei",
},
"headers": {
"host": "postman-echo.com",
"x-request-start": "t1742930191.318",
"connection": "close",
"x-forwarded-proto": "https",
"x-forwarded-port": "443",
"x-amzn-trace-id": "Root=1-67e3010f-4dc78cb667d6d4925b7d94b6",
"content-type": "text/plain",
"accept": "*/*",
"accept-language": "de-DE,de;q=0.5",
"user-agent": "Mozilla/4.0 (compatible; Win32; WinHttp.WinHttpRequest.5)"
},
"url": "https://postman-echo.com/get?foo1=hallo&foo2=doei"
}
```


Via een RESERVED WORD "dmoutput" gevolgd door het output-nummer, kan deze als bronformaat in een volgende element van de array wordt gebruikt. Omdat dit de 1e aanroep is, en daarmee het output-nummer 1 is, kan het JSON resultaat (verkregen via de GET) als brondata worden toegevoegd via RESERVED WORD "dmoutput1".

Stap 2 (bij stap 1 toevoegen): na het ophalen via **GET** van JSON formaat C uit systeem C (= Postman-echo.com), roepen we DataMixer aan met (bron) formaat A welke deze eenvoudige XML is:

```
<xml>
   <map>
      <code>Hello</code>
      <translate>Hallo</translate>
   </map>
   <map>
      <code>Hi</code>
      <translate>Hoi</translate>
   </map>
   <map>
      <code>Goodday</code>
      <translate>Goedendag</translate>
   </map>
</xml>
```

Het DM script levert een XML formaat B op die:

- De code uit XML formaat A moet opleveren waarvan de waarde in het veld 'translate' (case-insensitive) overeenkomt met de waarde van veld foo1 uit de JSON formaat C:

```
$start('DataMixer')
<xml>$value(//map$filter($upper(translate)=$upper(//foo1))/code)</xml>
$endstart
```

Dit levert als resultaat inderdaad de XML:

```
<xml> Hello</xml>
```

Om dit voor elkaar te krijgen, is het nodig om de webservice van DM via 2 sequentiële delen aan te roepen, d.w.z. eerst naar systeem C om de output

daarvan (JSON) te gebruiken als input voor het 2e deel. Let op gebruik van het RESERVED WORD "dmoutput1":

```
{
"DM":
[
{mb: {url: "https://postman-echo.com/get?foo1=hallo&foo2=doei", "method": "GET", "contenttype": "text/plain"}
},
{data: [  {"content":"$dm$output1"},
          {"content":
"<xml><map><code>Hello</code><translate>Hallo</translate></map><map><code>Hi</code><translate>Hoi</translate></map><map><code>Goodday</code><translate>Goededag</translate></map></xml>"}
      ],
      "dm$": $start('DataMixer')<xml>$value(//map$filter($upper(translate)=$upper(//foo1))/code)</xml>$endstart"
}

]
}
```

De structuur van een dergelijke aanroep met meerdere transacties verloopt dus via een array:

```
{
"DM":
[
{Transactie 1},
{Transactie 2}
]
}
```

Casussen

In de sectie worden een paar grotere casussen uitgewerkt. Vanuit de "store catalogue" als JSON brondata wordt de Shop Pricelist" doeldata gegenereerd in de formaten JSON, XML, CSV en HTML.

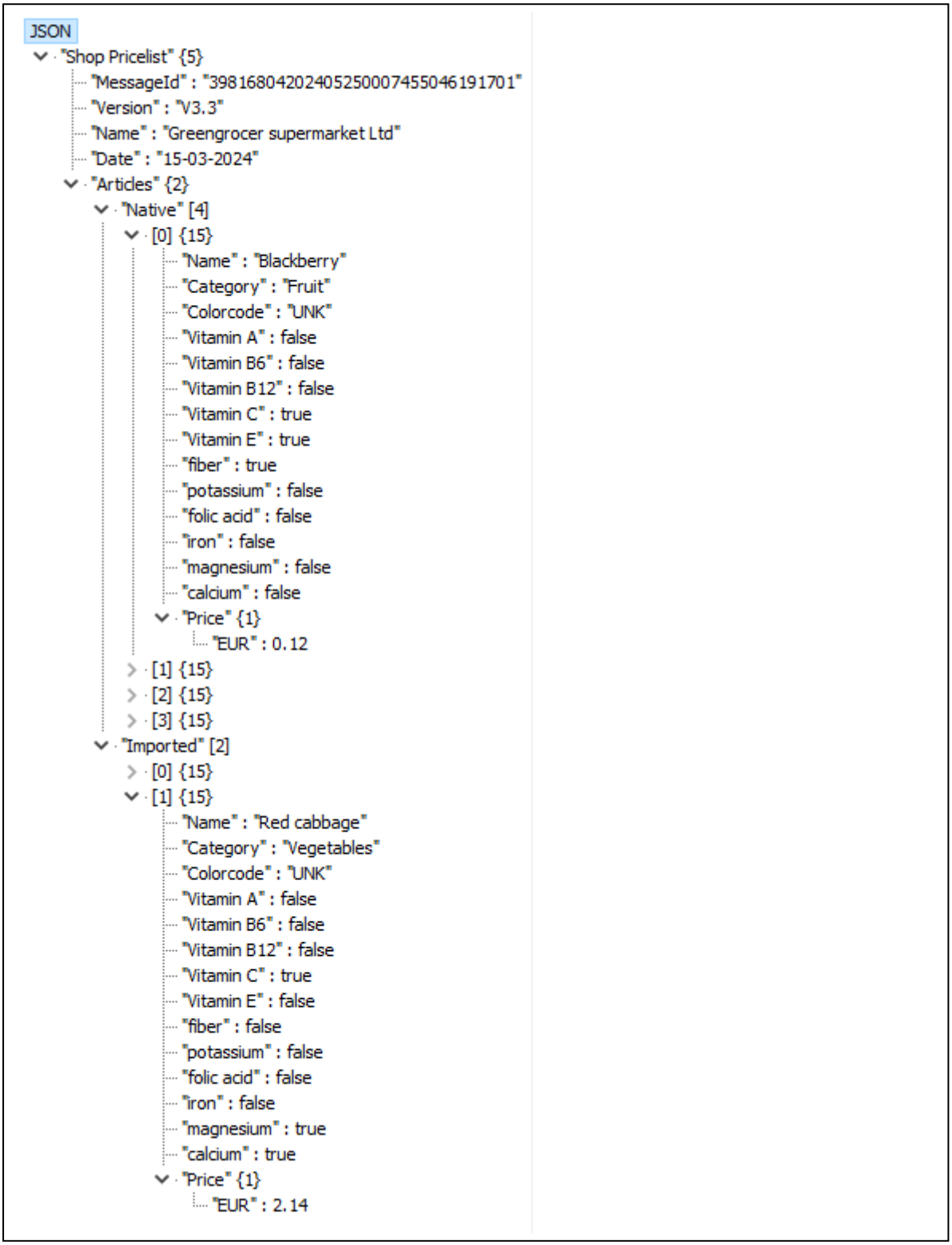
Er wordt in onderstaande casussen een wat uitgebreidere brondata gebruikt:

```
{"store catalogue":
 {
 "name": "Greengrocer supermarket Ltd",
 "version": 2.3,
 "date":"15-03-2024",
 "fruit":
  [
   { "name": "apple",
     "color": "red",
     "packaging unit" : "apiece",
     "weight": 0.26,
     "price": {"GBP": 0.50},
     "imported": false,
     "nutrients": ["fiber", "vitamin C", "potassium"] },
   { "name": "banana",
     "color": "yellow",
     "packaging unit" : "apiece",
     "weight": 0.18,
     "price": {"GBP": 0.30},
     "imported": true,
     "nutrients": ["fiber", "vitamin B6", "potassium"] },
   { "name": "strawberry",
     "color": "red",
     "packaging unit" : "container",
     "weight": 0.03,
     "price": {"GBP": 0.15},
     "imported": false,
     "nutrients": [ "vitamin C", "folic acid"] },
   { "name": "blackberry",
     "color": "black",
     "packaging unit" : "container",
     "weight": 0.01,
     "price": {"GBP": 0.10},
     "imported": false,
     "nutrients": ["fiber", "vitamin C", "vitamin E"] }
  ],
 "vegetables":
  [
   {  "name": "spinach",
     "color": "green",
     "packaging unit" : "jar",
     "weight": 0.30,
     "price": {"USD": 2.50},
     "imported": false,
     "nutrients": ["vitamin B6", "vitamin B12", "vitamin C", "vitamin D", "iron", "magnesium"] },
   {  "name": "red cabbage",
     "color": "purple",
     "packaging unit" : "jar",
     "weight": 0.30,
     "price": {"USD": 2.25},
     "imported": true,
     "nutrients": ["vitamin C", "vitamin D", "calcium", "magnesium"] }
  ]
 }
}
```

storecatalogue.json

JSON-JSON

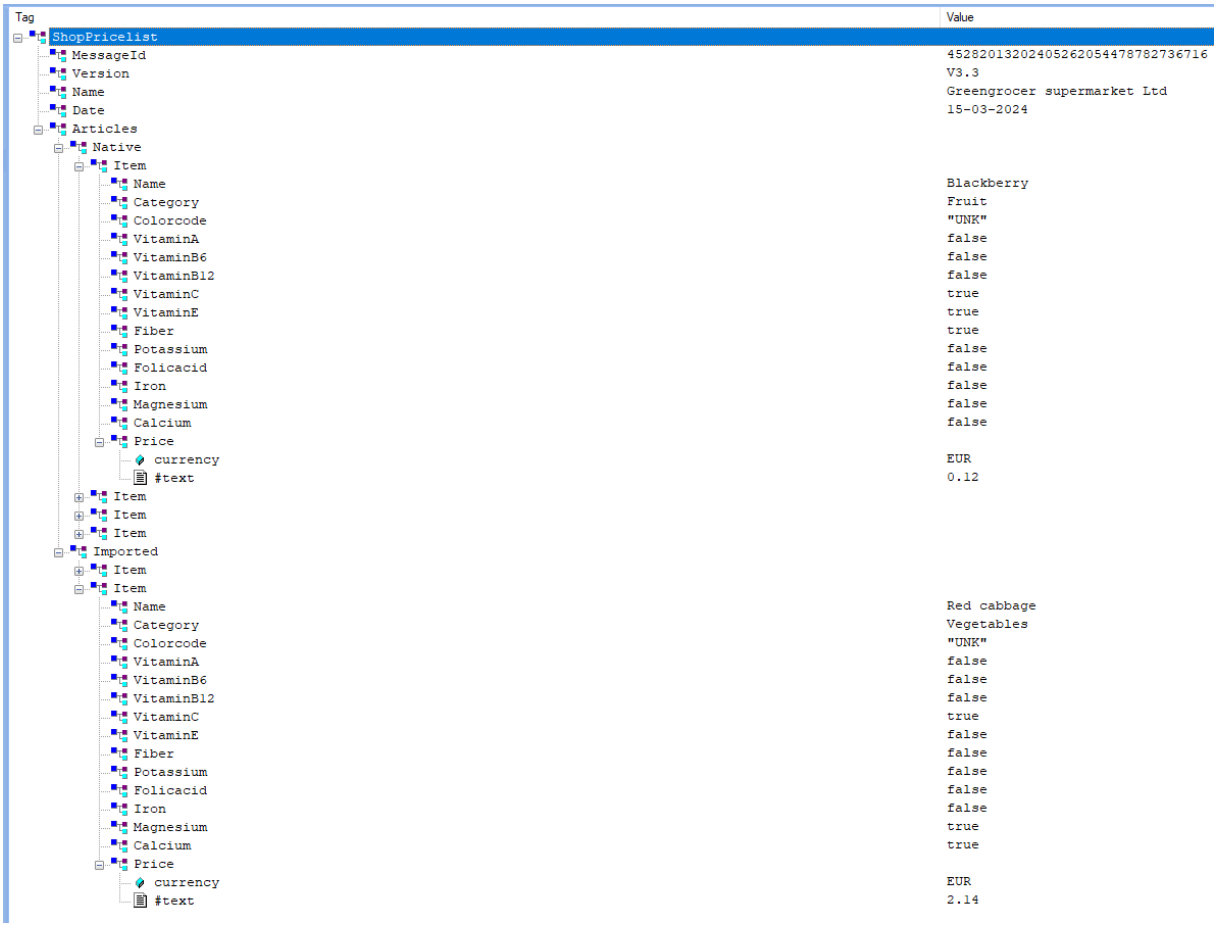
Doeldata na uitvoering van **storecatalogue_json_2_json.dcc** is beschikbaar in storecatalogue_json_2_json.json:



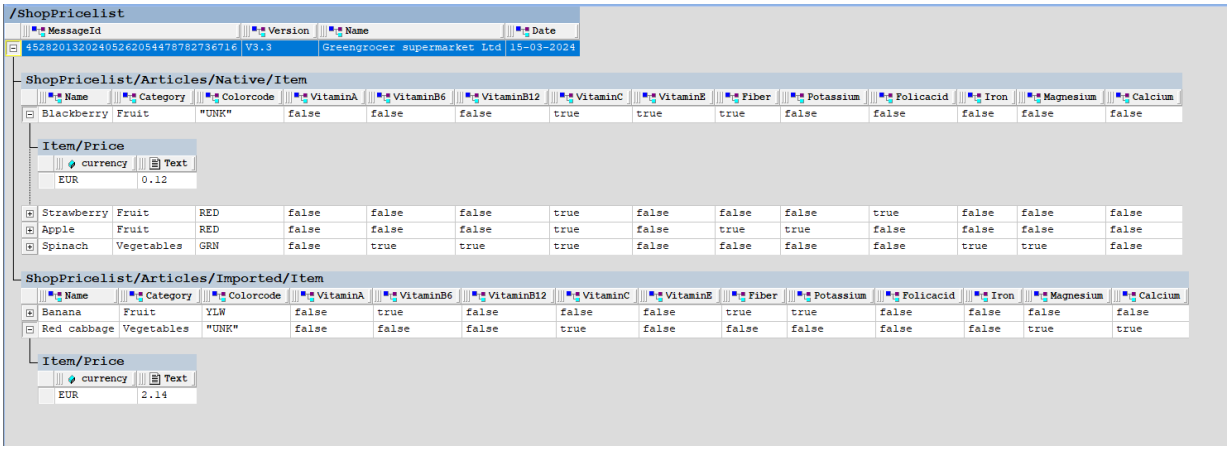
JSON-XML

De inhoud van de doeldata na uitvoering van **storecatalogue_json_2_xml.dcc** is beschikbaar in storecatalogue_json_2_xml.xml:


```xml
<ShopPricelist>
    <MessageId>45282013202405262054478782736716</MessageId>
    <Version>V3.3</Version>
    <Name>Greengrocer supermarket Ltd</Name>
    <Date>15-03-2024</Date>
    <Articles>
        <Native>
            <Item>
                <Name>Blackberry</Name>
                <Category>Fruit</Category>
                <Colorcode>"UNK"</Colorcode>
                <VitaminA>false</VitaminA>
                <VitaminB6>false</VitaminB6>
                <VitaminB12>false</VitaminB12>
                <VitaminC>true</VitaminC>
                <VitaminE>true</VitaminE>
                <Fiber>true</Fiber>
                <Potassium>false</Potassium>
                <Folicacid>false</Folicacid>
                <Iron>false</Iron>
                <Magnesium>false</Magnesium>
                <Calcium>false</Calcium>
                <Price currency="EUR">0.12</Price>
            </Item>
            <Item>
                <Name>Strawberry</Name>
                <Category>Fruit</Category>
                <Colorcode>RED</Colorcode>
                <VitaminA>false</VitaminA>
                <VitaminB6>false</VitaminB6>
                <VitaminB12>false</VitaminB12>
                <VitaminC>true</VitaminC>
                <VitaminE>false</VitaminE>
                <Fiber>false</Fiber>
                <Potassium>false</Potassium>
                <Folicacid>true</Folicacid>
                <Iron>false</Iron>
                <Magnesium>false</Magnesium>
                <Calcium>false</Calcium>
                <Price currency="EUR">0.17</Price>
            </Item>
            <Item>
                <Name>Apple</Name>
                <Category>Fruit</Category>
                <Colorcode>RED</Colorcode>
                <VitaminA>false</VitaminA>
                <VitaminB6>false</VitaminB6>
                <VitaminB12>false</VitaminB12>
                <VitaminC>true</VitaminC>
                <VitaminE>false</VitaminE>
                <Fiber>true</Fiber>
                <Potassium>true</Potassium>
                <Folicacid>false</Folicacid>
                <Iron>false</Iron>
                <Magnesium>false</Magnesium>
                <Calcium>false</Calcium>
                <Price currency="EUR">0.58</Price>
            </Item>
```



Tag	Value
ShopPricelist	
MessageId	45282013202405262054478782736716
Version	V3.3
Name	Greengrocer supermarket Ltd
Date	15-03-2024
Articles	
Native	
Item	
Name	Blackberry
Category	Fruit
Colorcode	"UNK"
VitaminA	false
VitaminB6	false
VitaminB12	false
VitaminC	true
VitaminE	true
Fiber	true
Potassium	false
Folicacid	false
Iron	false
Magnesium	false
Calcium	false
Price	
currency	EUR
#text	0.12
Item	
Item	
Item	
Imported	
Item	
Item	
Name	Red cabbage
Category	Vegetables
Colorcode	"UNK"
VitaminA	false
VitaminB6	false
VitaminB12	false
VitaminC	true
VitaminE	false
Fiber	false
Potassium	false
Folicacid	false
Iron	false
Magnesium	true
Calcium	true
Price	
currency	EUR
#text	2.14



/ShopPricelist

MessageId	Version	Name	Date
45282013202405262054478782736716	V3.3	Greengrocer supermarket Ltd	15-03-2024

ShopPricelist/Articles/Native/Item

Name	Category	Colorcode	VitaminA	VitaminB6	VitaminB12	VitaminC	VitaminE	Fiber	Potassium	Folicacid	Iron	Magnesium	Calcium
Blackberry	Fruit	"UNK"	false	false	false	true	true	true	false	false	false	false	false
Strawberry	Fruit	RED	false	false	false	true	false	false	false	true	false	false	false
Apple	Fruit	RED	false	false	false	true	false	true	true	false	false	false	false
Spinach	Vegetables	GRN	false	true	true	true	false	false	false	false	true	true	false

Item/Price

currency	Text
EUR	0.12

ShopPricelist/Articles/Imported/Item

Name	Category	Colorcode	VitaminA	VitaminB6	VitaminB12	VitaminC	VitaminE	Fiber	Potassium	Folicacid	Iron	Magnesium	Calcium
Banana	Fruit	YLW	false	true	false	false	false	true	true	false	false	false	false
Red cabbage	Vegetables	"UNK"	false	false	false	true	false	false	false	false	false	true	true

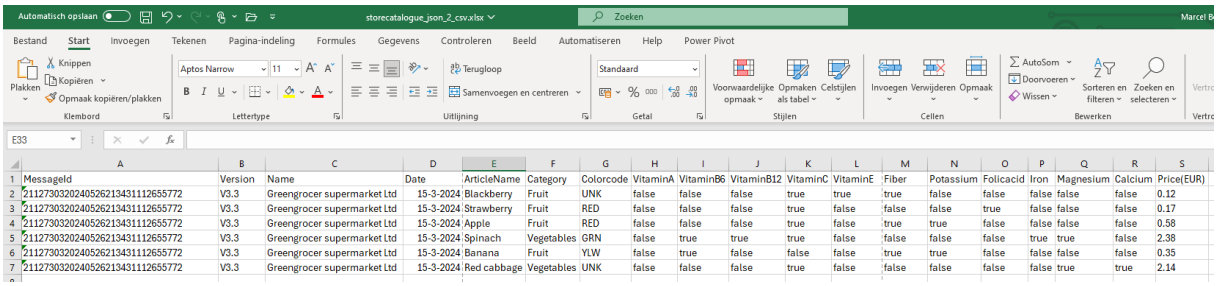
Item/Price

currency	Text
EUR	2.14

JSON-CSV

De inhoud van de doeldata na uitvoering van **storecatalogue_json_2_csv.dcc** is beschikbaar in storecatalogue_json_2_csv.csv:

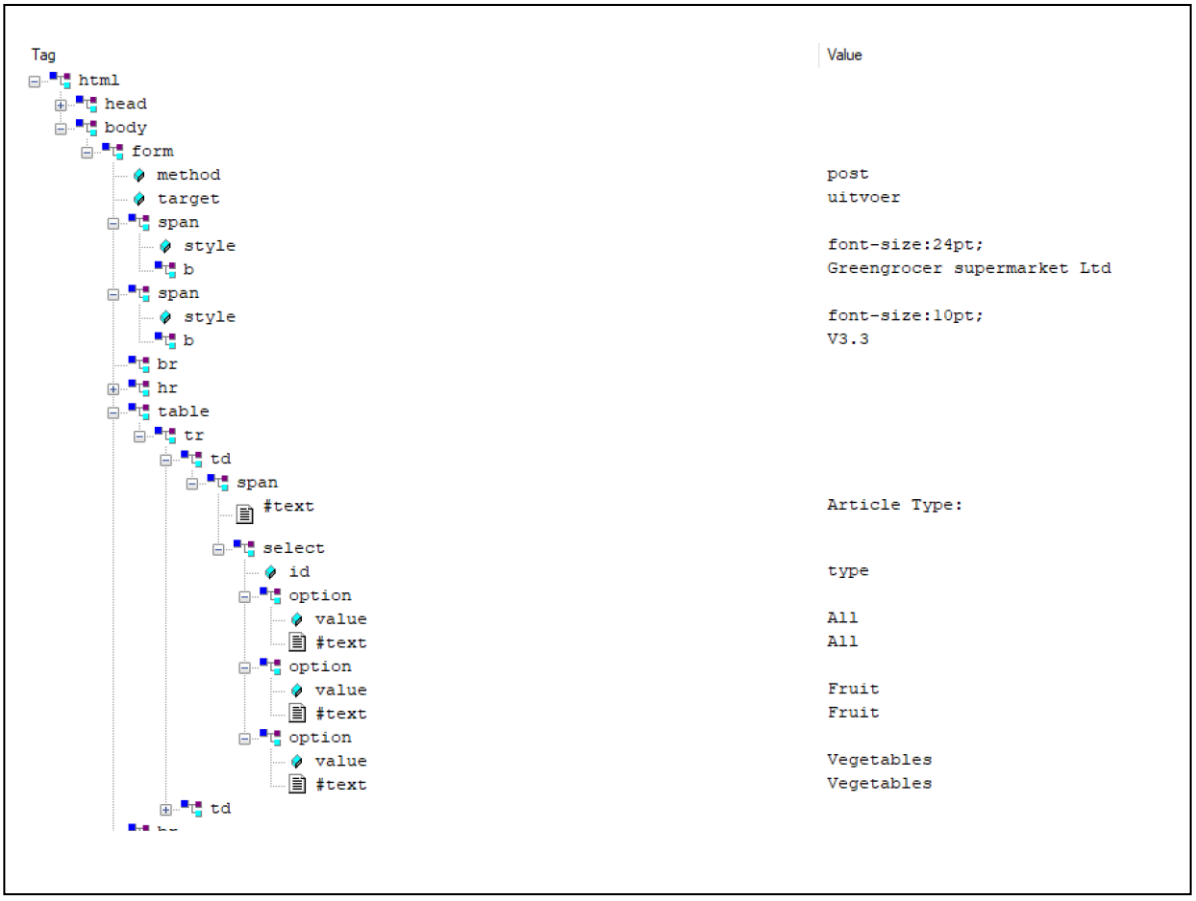
```
MessageId;Version;Name;Date;ArticleName;Category;Colorcode;VitaminA;VitaminB6;VitaminB12;VitaminC;VitaminE;Fiber;Potassium;Folicacid;Iron;Magnesium;Calcium;Price(EUR);
21127303202405262134311126557 72;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Blackberry;Fruit;UNK;false;false;false;true;true;true;false;false;false;false;false;0.12;
21127303202405262134311126557 72;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Strawberry;Fruit;RED;false;false;false;true;false;false;false;true;false;false;false;0.17;
21127303202405262134311126557 72;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Apple;Fruit;RED;false;false;false;true;false;true;true;false;false;false;false;0.58;
21127303202405262134311126557 72;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Spinach;Vegetables;GRN;false;true;true;true;false;false;false;false;true;true;false;2.38;
21127303202405262134311126557 72;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Banana;Fruit;YLW;false;true;false;false;false;true;true;false;false;false;false;0.35;
21127303202405262134311126557 72;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Red cabbage;Vegetables;UNK;false;false;false;true;false;false;false;false;false;true;true;2.14;
```



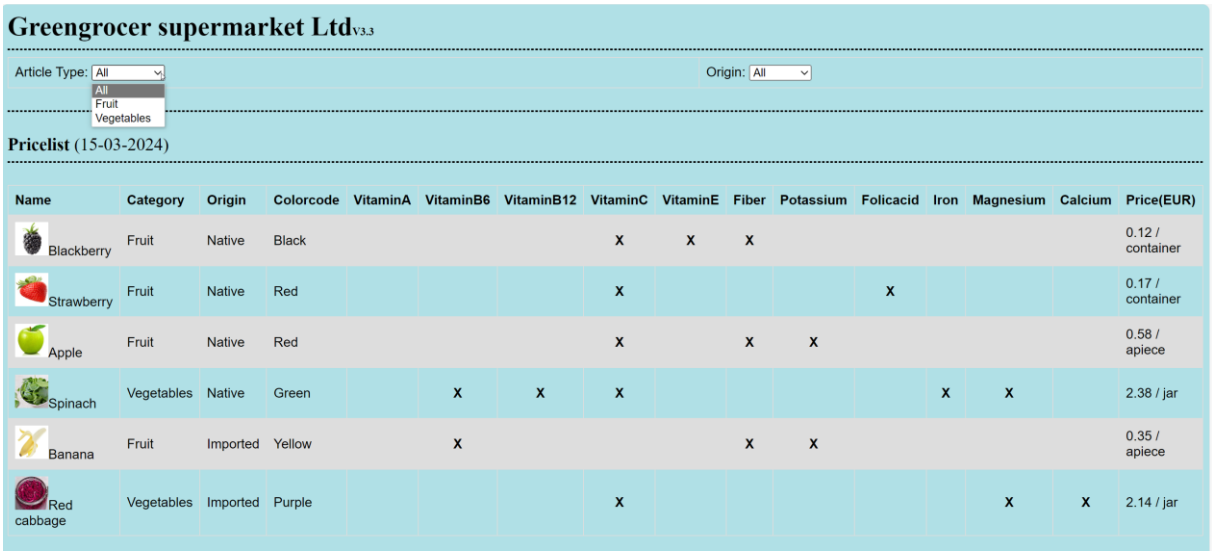
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	MessageId	Version	Name	Date	ArticleName	Category	Colorcode	VitaminA	VitaminB6	VitaminB12	VitaminC	VitaminE	Fiber	Potassium	FolicaId	Iron	Magnesium	Calcium	Price(EUR)
2	21127303202405262134311126557 72	V3.3	Greengrocer supermarket Ltd	15-3-2024	Blackberry	Fruit	UNK	false	false	false	true	true	true	false	false	false	false	false	0.12
3	21127303202405262134311126557 72	V3.3	Greengrocer supermarket Ltd	15-3-2024	Strawberry	Fruit	RED	false	false	false	true	false	false	false	true	false	false	false	0.17
4	21127303202405262134311126557 72	V3.3	Greengrocer supermarket Ltd	15-3-2024	Apple	Fruit	RED	false	false	false	true	false	true	true	false	false	false	false	0.58
5	21127303202405262134311126557 72	V3.3	Greengrocer supermarket Ltd	15-3-2024	Spinach	Vegetables	GRN	false	true	true	true	false	false	false	false	true	true	false	2.38
6	21127303202405262134311126557 72	V3.3	Greengrocer supermarket Ltd	15-3-2024	Banana	Fruit	YLW	false	true	false	false	false	true	true	false	false	false	false	0.35
7	21127303202405262134311126557 72	V3.3	Greengrocer supermarket Ltd	15-3-2024	Red cabbage	Vegetables	UNK	false	false	false	true	false	false	false	false	false	true	true	2.14

JSON-HTML

De inhoud van de doeldata na uitvoering van **storecatalogue_json_2_htm.dcc** is beschikbaar in storecatalogue_json_2_htm.htm:



Laden van de HTML doeldata in een internet-browser geeft het volgende resultaat:



Greengrocer supermarket Ltd v3.3

Article Type: All (All, Fruit, Vegetables) Origin: All

Pricelist (15-03-2024)

Name	Category	Origin	Colorcode	VitaminA	VitaminB6	VitaminB12	VitaminC	VitaminE	Fiber	Potassium	Folicacid	Iron	Magnesium	Calcium	Price(EUR)
Blackberry	Fruit	Native	Black				X	X	X						0.12 / container
Strawberry	Fruit	Native	Red				X				X				0.17 / container
Apple	Fruit	Native	Red				X		X	X					0.58 / apiece
Spinach	Vegetables	Native	Green		X	X	X					X	X		2.38 / jar
Banana	Fruit	Imported	Yellow		X				X	X					0.35 / apiece
Red cabbage	Vegetables	Imported	Purple				X						X	X	2.14 / jar

XML-JSON

We gaan nu reverse vanuit de verkregen/gegenereerde XML "storecatalogue_json_2_xml.xml" als doeldata de hierboven gebruikte JSON (storecatalogue.json) weer terug genereren.

De inhoud van deze doeldata na uitvoering van **storecatalogue_xml_2_json.dcc** is beschikbaar in storecatalogue_xml_2_json.json (en is vrijwel gelijk aan storecatalogue.json):

```
<ShopPricelist>
    <MessageId>45282013202405262054478782736716</MessageId>
    <Version>V3.3</Version>
    <Name>Greengrocer supermarket Ltd</Name>
    <Date>15-03-2024</Date>
    <Articles>
        <Native>
            <Item>
                <Name>Blackberry</Name>
                <Category>Fruit</Category>
                <Colorcode>"UNK"</Colorcode>
                <VitaminA>false</VitaminA>
                <VitaminB6>false</VitaminB6>
                <VitaminB12>false</VitaminB12>
                <VitaminC>true</VitaminC>
                <VitaminE>true</VitaminE>
                <Fiber>true</Fiber>
                <Potassium>false</Potassium>
                <Folicacid>false</Folicacid>
                <Iron>false</Iron>
                <Magnesium>false</Magnesium>
                <Calcium>false</Calcium>
                <Price currency="EUR">0.12</Price>
            </Item>
            <Item>
                <Name>Strawberry</Name>
                <Category>Fruit</Category>
                <Colorcode>RED</Colorcode>
                <VitaminA>false</VitaminA>
                <VitaminB6>false</VitaminB6>
                <VitaminB12>false</VitaminB12>
                <VitaminC>true</VitaminC>
                <VitaminE>false</VitaminE>
                <Fiber>false</Fiber>
                <Potassium>false</Potassium>
                <Folicacid>true</Folicacid>
                <Iron>false</Iron>
                <Magnesium>false</Magnesium>
                <Calcium>false</Calcium>
                <Price currency="EUR">0.17</Price>
            </Item>
            <Item>
                <Name>Apple</Name>
                <Category>Fruit</Category>
                <Colorcode>RED</Colorcode>
                <VitaminA>false</VitaminA>
                <VitaminB6>false</VitaminB6>
                <VitaminB12>false</VitaminB12>
                <VitaminC>true</VitaminC>
                <VitaminE>false</VitaminE>
                <Fiber>true</Fiber>
                <Potassium>true</Potassium>
                <Folicacid>false</Folicacid>
                <Iron>false</Iron>
                <Magnesium>false</Magnesium>
                <Calcium>false</Calcium>
                <Price currency="EUR">0.58</Price>
            </Item>
```

storecatalogue_json_2_xml.xml

```
{"store catalogue":
 {
 "name": "Greengrocer supermarket Ltd",
 "version": 2.3,
 "date": "15/03/2024",
 "fruit": [
{"name": "apple",
 "color": "red",
 "price":{"GBP": 0.5},
 "imported": false,
 "nutrients": ["vitamin C","fiber","potassium"]
 }
,
 {"name": "blackberry",
 "color":  "Unknown" ,
 "price":{"GBP": 0.1},
 "imported": false,
 "nutrients": ["vitamin C","vitamin E","fiber"]
 }
,
 {"name": "strawberry",
 "color": "red",
 "price":{"GBP": 0.15},
 "imported": false,
 "nutrients": ["vitamin C","folic acid"]
 }

        ]
        ,
 "vegetables": [
{"name": "spinach",
 "color": "green",
 "price":{"USD": 2.51},
 "imported": true,
 "nutrients": ["vitamin B6","vitamin B12","vitamin C","iron","magnesium"]
 }

        ]
 }
}
```


storecatalogue_xml_2_json.json

```
{
    "store catalogue": {
        "name": "Greengrocer supermarket Ltd",
        "version": 2.3,
        "date": "15/03/2024",
        "fruit": [
            {
                "name": "apple",
                "color": "red",
                "price": {
                    "GBP": 0.5
                },
                "imported": false,
                "nutrients": [
                    "vitamin C",
                    "fiber",
                    "potassium"
                ]
            },
            {
                "name": "blackberry",
                "color": "Unknown",
                "price": {
                    "GBP": 0.1
                },
                "imported": false,
                "nutrients": [
                    "vitamin C",
                    "vitamin E",
                    "fiber"
                ]
            },
            {
                "name": "strawberry",
                "color": "red",
                "price": {
                    "GBP": 0.15
                },
                "imported": false,
                "nutrients": [
                    "vitamin C",
                    "folic acid"
                ]
            }
        ],
        "vegetables": [
            {
                "name": "spinach",
                "color": "green",
                "price": {
                    "USD": 2.51
                },
                "imported": true,
                "nutrients": [
                    "vitamin B6",
                    "vitamin B12",
                    "vitamin C",
                    "iron",
                    "magnesium"
                ]
            }
        ]
    }
}
```

XML-XML

We gaan nogmaals vanuit de verkregen/gegenereerde XML "storecatalogue_json_2_xml.xml" als doeldata een XML genereren die qua structuur gelijk is aan de gegenereerde JSON (storecatalogue.json) uit de vorige sectie.

De inhoud van deze doeldata na uitvoering van **storecatalogue_xml_2_xml.dcc** is beschikbaar in storecatalogue_xml_2_xml.xml (en is qua structuur vrijwel gelijk aan storecatalogue.json). Let op het gegenereerde namespace (in de XML tag storecatalogue), hetgeen mogelijk is via functie $namespace:

```
<storecatalogue xmlns="http://www.grocerycatalogueformat.com">
  <name>Greengrocer supermarket Ltd</name>
  <version>2.3</version>
  <date>15/03/2024</date>
  <fruit>
    <item>
      <name>apple</name>
      <color>red</color>
      <price curr="GBP">0.5</price>
      <imported>false</imported>
      <nutrients>
        <nutrient>vitamin C</nutrient>
        <nutrient>fiber</nutrient>
        <nutrient>potassium</nutrient>
      </nutrients>
    </item>
    <item>
      <name>blackberry</name>
      <color>Unknown</color>
      <price curr="GBP">0.1</price>
      <imported>false</imported>
      <nutrients>
        <nutrient>vitamin C</nutrient>
        <nutrient>vitamin E</nutrient>
        <nutrient>fiber</nutrient>
      </nutrients>
    </item>
    <item>
      <name>strawberry</name>
      <color>red</color>
      <price curr="GBP">0.15</price>
      <imported>false</imported>
      <nutrients>
        <nutrient>vitamin C</nutrient>
        <nutrient>folic acid</nutrient>
      </nutrients>
    </item>
  </fruit>
  <vegetables>
    <item>
      <name>spinach</name>
      <color>green</color>
      <price curr="USD">2.51</price>
      <imported>true</imported>
      <nutrients>
        <nutrient>vitamin B6</nutrient>
        <nutrient>vitamin B12</nutrient>
        <nutrient>vitamin C</nutrient>
        <nutrient>iron</nutrient>
        <nutrient>magnesium</nutrient>
      </nutrients>
    </item>
  </vegetables>
</storecatalogue>
```

storecatalogue_xml_2_xml.xml

XML-CSV

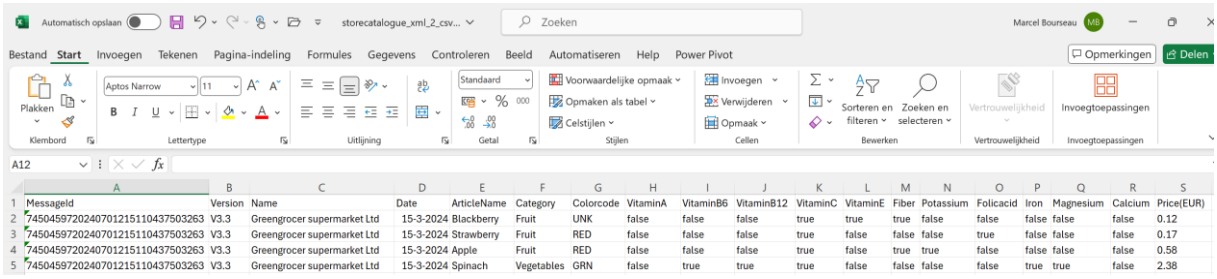
We gaan als brondata uit van de gegenereerde "storecatalogue_xml_2_xml.xml" uit de casus XML-XML, om vervolgens als doeldata dezelfde CSV te genereren uit de JSON-CSV casus. Hierbij is de functie $namespace nodig om de XML uit te kunnen lezen.

```
$start('DataMixer')
$namespaces
xmlns="http://www.grocerycatalogueformat.com"
$endnamespaces
MessageId;Version;Name;Date;ArticleName;Category;Colorcode;VitaminA;VitaminB6;VitaminB12;VitaminC;VitaminE;Fiber;Potassium;Folicacid;Iron;Magnesium;Calcium;Price(EUR);
$loop(//storecatalogue/fruit/item$filter(imported='false'))$orderby(price, num, asc)
$ArticleData(articletype='fruit')
$endloop
$loop(//storecatalogue/vegetables/item$filter(imported='false'))$orderby(price, num, asc)
$ArticleData(articletype='vegetables')
$endloop
$loop(//storecatalogue/fruit/item$filter(imported='true'))$orderby(price, num, asc)
$ArticleData(articletype='fruit')
$endloop
$loop(//storecatalogue/vegetables/item$filter(imported='true'))$orderby(price, num, asc)
$ArticleData(articletype='vegetables')
$endloop
$endstart

$/* Exchange rate to EUR of GBR and USD $*/
$var(gbp_2_euro_exchange_rate = 1.16,usd_2_euro_exchange_rate = 0.95)

$/* Does value x exists in array y
e.g. x=2 and y=[1,2,3] -> true
and  x=4 and y=[1,2,3] -> false
$*/
$function(ExistsInNamelessArray)
$parameter(x)
$parameter(y)
$value($count($array($y)$filter($upper(.)=$upper($x))) > 0)
$endfunction

$/* Color translation into 3 char code $*/
$function(TranslateColor)
$parameter(x)
  $case
    $when ($x = 'red')RED$endwhen
    $when ($x = 'yellow')YLW$endwhen
    $when ($x = 'green')GRN$endwhen
    $when ($x = 'blue')BLU$endwhen
    $when ($x = 'orange')ORA$endwhen
    $when ($x = 'pink')PNK$endwhen
    $defaultUNK$enddefault
  $endcase
$endfunction

$function(CapsFirstCharOfString)
$parameter(x)
$value($upper($substr($x,1,1)))$value($substr($x,2,$strlen($x)))
$endfunction


$/* Trunc real to max. of 2 decimals $*/
$function(trunc_2_decimals)
$parameter(x)
  $value($div($round($x * 100),100))
$endfunction

$function(ArticleData)
$parameter(articletype)
$var(vitA='Vitamin$spA')
$var(vitB6='Vitamin$spB6')
$var(vitB12='Vitamin$spB12')
$var(vitC='Vitamin$spC')
$var(vitE='Vitamin$spE')
$var(fiber='fiber')
$var(potassium='potassium')
$var(folac='folic$spacid')
$var(fe='iron')
$var(mg='magnesium')
$var(calc='calcium')
$value($messageid);V$value(//storecatalogue/version + 1);"$CapsFirstCharOfString(x=//storecatalogue/name)";$value($replace(//storecatalogue/date, '/', '-
'));$CapsFirstCharOfString(x=name);$CapsFirstCharOfString(x=$articletype);$TranslateColor(x=color);$ExistsInNamelessArray(x=$vitA, y=nutrients);$ExistsInNamelessArray(x=$vitB6,
y=nutrients);$ExistsInNamelessArray(x=$vitB12, y=nutrients);$ExistsInNamelessArray(x=$vitC, y=nutrients);$ExistsInNamelessArray(x=$vitE, y=nutrients);$ExistsInNamelessArray(x=$fiber,
y=nutrients);$ExistsInNamelessArray(x=$potassium, y=nutrients);$ExistsInNamelessArray(x=$folac, y=nutrients);$ExistsInNamelessArray(x=$fe, y=nutrients);$ExistsInNamelessArray(x=$mg,
y=nutrients);$ExistsInNamelessArray(x=$calc, y=nutrients);$case$when (price$filter(@curr='GBP') != '')$trunc_2_decimals(x=$gbp_2_euro_exchange_rate *
price$filter(@curr='GBP'))$endwhen$when (price$filter(@curr='USD') != '')$trunc_2_decimals(x=$usd_2_euro_exchange_rate * price$filter(@curr='USD'))$endwhen$endcase;
$endfunction
```

storecatalogue_xml_2_csv.dcc

De inhoud van de doeldata na uitvoering van **storecatalogue_xml_2_csv.dcc** is beschikbaar in storecatalogue_xml_2_csv.csv:

```
MessageId;Version;Name;Date;ArticleName;Category;Colorcode;VitaminA;VitaminB6;VitaminB12;VitaminC;VitaminE;Fiber;Potassium;Folicacid;Iron;Magnesium;Calcium;Price(EUR);
74504597202407012151104375032 63;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Blackberry;Fruit;UNK;false;false;false;true;true;true;false;false;false;false;false;0.12;
74504597202407012151104375032 63;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Strawberry;Fruit;RED;false;false;false;true;false;false;false;true;false;false;false;0.17;
74504597202407012151104375032 63;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Apple;Fruit;RED;false;false;false;true;false;true;true;false;false;false;false;0.58;
74504597202407012151104375032 63;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Spinach;Vegetables;GRN;false;true;true;true;false;false;false;false;true;true;false;2.38;
```



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	MessageId	Version	Name	Date	ArticleName	Category	Colorcode	VitaminA	VitaminB6	VitaminB12	VitaminC	VitaminE	Fiber	Potassium	Folicacid	Iron	Magnesium	Calcium	Price(EUR)
2	74504597202407012151104375032 63	V3.3	Greengrocer supermarket Ltd	15-3-2024	Blackberry	Fruit	UNK	false	false	false	true	true	true	false	false	false	false	false	0.12
3	74504597202407012151104375032 63	V3.3	Greengrocer supermarket Ltd	15-3-2024	Strawberry	Fruit	RED	false	false	false	true	false	false	false	true	false	false	false	0.17
4	74504597202407012151104375032 63	V3.3	Greengrocer supermarket Ltd	15-3-2024	Apple	Fruit	RED	false	false	false	true	false	true	true	false	false	false	false	0.58
5	74504597202407012151104375032 63	V3.3	Greengrocer supermarket Ltd	15-3-2024	Spinach	Vegetables	GRN	false	true	true	true	false	false	false	false	true	true	false	2.38

CSV-JSON met header

We gaan nu vanuit de gegenereerde CSV "storecatalogue_json_2_csv.csv", die we in dit voorbeeld als doeldata gebruiken, een XML genereren.

```
MessageId;Version;Name;Date;ArticleName;Category;Colorcode;VitaminA;VitaminB6;VitaminB12;VitaminC;VitaminE;Fiber;Potassium;Folicacid;Iron;Magnesium;Calcium;Price(EUR);
74504597202407012151104375032 63;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Blackberry;Fruit;UNK;false;false;false;true;true;true;false;false;false;false;false;0.12;
74504597202407012151104375032 63;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Strawberry;Fruit;RED;false;false;false;true;false;false;false;true;false;false;false;0.17;
74504597202407012151104375032 63;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Apple;Fruit;RED;false;false;false;true;false;true;true;false;false;false;false;0.58;
74504597202407012151104375032 63;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Spinach;Vegetables;GRN;false;true;true;true;false;false;false;false;true;true;false;2.38;
```

storecatalogue_json_2_csv.csv

```
$start('DataMixer')
<ShopPricelist>
<MessageId>$value(//csv/MessageId)</MessageId>
<Version>$value(//csv/Version)</Version>
<Name>$value(//csv/Name)</Name>
<Date>$value(//csv/Date)</Date>
<Articles>
$loop(//csv)
<Item>
        <Category>$value(Category)</Category>
        <Name>$value(ArticleName)</Name>
</Item>
$endloop
</Articles>
</ShopPricelist>
$endstart
```

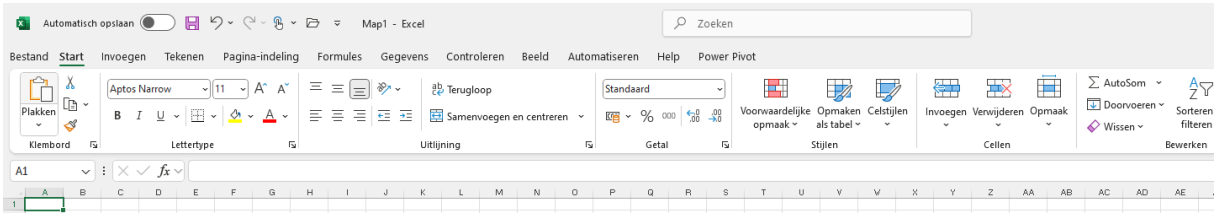
Dit resulteert in deze XML:

```
<ShopPricelist>
<MessageId>74504597202407012151104375032 63</MessageId>
<Version>V3.3</Version>
<Name>"Greengrocer supermarket Ltd"</Name>
<Date>15-03-2024</Date>
<Articles>
<Item>
<Category>Fruit</Category>
<Name>Blackberry</Name>
</Item>
<Item>
<Category>Fruit</Category>
<Name>Strawberry</Name>
</Item>
<Item>
<Category>Fruit</Category>
<Name>Apple</Name>
</Item>
<Item>
<Category>Vegetables</Category>
<Name>Spinach</Name>
</Item>
</Articles>
</ShopPricelist>
```

CSV-JSON zonder header

In vorige voorbeeld werd een CSV bestand gebruikt met een header (kop). De waarden uit de CSV zijn te benaderen via b.v. de functie $value door daarin de headernaam te gebruiken, zoals b.v. $value(MessageId).

In geval dat de CSV geen header (kop) heeft, maar dat de waarden al beginnen op regel 1, kan gebruik worden gemaakt van de standaard Excel kolomnamen: A t/m Z, AA t/m AZ, BA t/m BZ etc:



Om gebruik te kunnen maken van de standaard Excel kolomnamen, is het wel nodig om dit in de API-aanroep op te geven. Daartoe moet dan de optionele veld “options” worden meegestuurd met daarin de csv en header attributen. Door de header de waarde 0 te geven (of waarde false is ook toegestaan), is het mogelijk

```
{
        "DM": {
                "data": [
                        {
                                "content": "….hier staat de brondata 1…",
                                "options": {
                                        "csv": {
                                                "header": false
                                        }
                                }
                        }
                ],
                "dm$": "… hier staat DM $script om de brondata te converteren …  "
        }
}
```

om de standaard Excelkolomnamen te gebruiken.

(*) Wanneer options.csv.header niet wordt meegeven, dan wordt ervan uitgegaan dan de CSV een header heeft (defaultwaarde = 1 of true), en kunnen de standaard Excelkolomnamen niet worden toegepast, maar dienen de header kolomnamen te worden gebruikt.

In geval van deze CSV waarbij er geen header is…

```
745045972024070121511043750326 3;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Blackberry;Fruit;UNK;false;false;false;true;true;true;false;false;false;false;false;0.12;
745045972024070121511043750326 3;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Strawberry;Fruit;RED;false;false;false;true;false;false;false;true;false;false;false;0.17;
745045972024070121511043750326 3;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Apple;Fruit;RED;false;false;false;true;false;true;true;false;false;false;false;0.58;
745045972024070121511043750326 3;V3.3;"Greengrocer supermarket Ltd";15-03-2024;Spinach;Vegetables;GRN;false;true;true;true;false;false;false;false;true;true;true;false;2.38;
```

… kunnen met de options "csv.header" = true de standaard kolomnamen (in hoofdletters) worden gebruikt…

```
$start('DataMixer')
<ShopPricelist>
<MessageId>$value(//csv/A)</MessageId>
<Version>$value(//csv/B)</Version>
<Name>$value(//csv/C)</Name>
<Date>$value(//csv/D)</Date>
<Articles>
$loop(//csv)
<Item>
        <Category>$value(F)</Category>
        <Name>$value(E)</Name>
</Item>
$endloop
</Articles>
</ShopPricelist>
$endstart
```

… wat eveneens hetzelfde resultaat geeft als de output van de vorige paragraaf.

```
<ShopPricelist>
<MessageId>745045972024070121511043750326 3</MessageId>
<Version>V3.3</Version>
<Name>"Greengrocer supermarket Ltd"</Name>
<Date>15-03-2024</Date>
<Articles>
<Item>
<Category>Fruit</Category>
<Name>Blackberry</Name>
</Item>
<Item>
<Category>Fruit</Category>
<Name>Strawberry</Name>
</Item>
<Item>
<Category>Fruit</Category>
<Name>Apple</Name>
</Item>
<Item>
<Category>Vegetables</Category>
<Name>Spinach</Name>
</Item>
</Articles>
</ShopPricelist>
```

Referentie Gids

In dit hoofdstuk wordt een overzicht van alle DM functies alfabetisch gesorteerd weergegeven:

Voorbeeld Brondata

{"store catalogue":
{
"name": "Greengrocer supermarket Ltd",
"version": 2.3,
"date": "15-03-2024",
"fruit":
[
{
"name": "apple",
"color": "red",
"weight": 0.26
},
{
"name": "banana",
"color": "yellow",
"weight": 0.18
},
{
"name": "strawberry",
"color": "red",
"weight": 0.03
}
]
}
}

Categorie: Arithmetic

DM functie	Aanroep	Effect/doel	Voorbeeld	Resultaat
+	number + number	Som van twee getallen	$value(5+//version)	7.3
-	number - number	Verschil tussen twee getallen	$value(5-//version)	2.7
*	number * number	Product van twee getallen	$value(3*2)	11.5
$ceil	$ceil(number)	Afronden van getal naar boven	$value(**$ceil**(//version))	3
$count	$count(folder)	Tellen van het aantal velden uit de folder	$value(**$count**(//name))	4
$div	$div(number, number)	Quotient van twee getallen	$value(**$div**(5,//version))	2.17391304347826
$floor	$floor(number)	Afronden van getal naar beneden	$value(**$floor**(//version))	2
$messageid	$messageid	Random messageID van 30 cijfers	$value(**$messageid**)	5332...7298
$random	$random	Random getal tussen 0 en 1	$value(**$random**)	0.775503696531374
$round	$round(number)	Afronden van getal naar boven of onder	$value(**$round**(//version))	2
$sum	$sum(folder)	Optellen van alle velden uit de folder-expressie	$value(**$sum**(//weight))	0.47

Categorie: Constructing

$elem	$elem(string, string) DM$cript $endelem	Creeer XML/HTML tag genaamd 'string', 2e string is optioneel voor namespace	**$elem**('span')rood**$endelem**	rood
$elemname				
$outputistxt				
$outputisxml				
$putattr	$putattr(string) DM$cript $endelem	Creeer XML/HTML attribuut genaamd 'string', i.c.m. $elem te gebruiken	$elem('span')**$putattr**('style')color:$value(//color)**$endputattr**rood$endelem	rood

Categorie: Datums

Data	$isodate	$isodate	Huidige datum/tijdstip in ISO formaat	$value(**$isodate**)	2025-02-08T16:59:44Z

Categorie: Definition

$/* $*/	$/* ... $*/	Commentaar m.b.t. je DM$cript	**$/*** Voorbeeld **$*/**	-
$comment	$comment .... $endcomment	Commentaar m.b.t. je DM$cript	**$comment** Voorbeeld **$endcomment**	-
$function	$function(name) DM$cript $endfunction	Definitie van een eigen functie genaamd 'name'	**$function**(BepaalJaar)$value($substr($isodate,1,4))**$endfunction**	$BepaalJaar() resulteert in: 2025
$iref	$iref(iref-code)	Includen van intern data(bestand) obv iref-code	**$iref**(colortable)$value(//colortable/Row$filter(Kleur='Rood')/Hex)	#ff0000
$parameter	$parameter(name)	Definitie bij een eigen functie van een parameter genaamd 'name'	$function(BepaalJaarPlus)**$parameter**(x)$value($substr($isodate,1,4)+$x)$endfunction	$BepaalJaarPlus(x=3) resulteert in: 2028
$start	$start(string) DM$cript $endstart	Startpunt en eindpunt van DM$cript	**$start**('DataMixer') .... **$endstart**	
$var	$var(name = value)	Initialiseren van een variabele genaamd 'name' met waarde 'value'	**$var**(y=$value($substr(//date,7,4)))$BepaalJaarPlus(x=$y)	$var(y=2)$BepaalJaarPlus(x=$y) resulteert in 4049
$vardata				
$xref	$xref(xref-code)	Includen van extern data(bestand) obv xref-code	**$xref**(eur2usd) $value(//eur2usd//cb:value)	1,0377

Categorie: Error handling

$error	$error(number, string)	Genereert een HTTP status (error) met code 'number', gecombineerd met de error-tekst 'string', en retourneert dit als resultaat van DM$cript	**$error**(400,'Voorbeeld$spvan$speen$spfout')	HTTP status code 400 wordt als response teruggestuurd naar aanroeper met HTTP response tekst {"DM$_RUNTIME_ERROR" : "'Voorbeeld van een fout'"}
$httpstatus	$httpstatus(number)	Genereert een HTTP status (error) met code 'number', gecombineerd met het resultaat van DM$script wat als resultaat wordt geretourneerd	**$httpstatus**(400)	HTTP status code 400 wordt als response teruggestuurd naar aanroeper met HTTP response tekst het resultaat van DM$cript

Categorie: Logical programming

$coalesce	$coalesce(folder expr, folder expr,...)	De eerste folder expressie waarvan de waarde bestaat c.q. niet leeg is, is het resultaat van deze functie $coalesce	$coalesce(//versionnumber,//version,//catalogue version)	2.3
$filter	$filter(conditie expressie)	Een deel van DM$cript wordt uitgevoerd als aan voorwaarde 'conditie expression' wordt voldaan	$loop($array(//fruit)**$filter**(color='red')) <naam>$value(name)</naam> $endloop	<naam>apple</naam> <naam>strawberry</naam>
$groupby	$groupby(DM folder expressie)	I.c.m. $filter te gebruiken: om de loop te groeperen op 1 specifiek veld	$loop($array(//fruit)**$filter(**$groupby**(**color**)**)) <kleur>$value(color)</kleur> $endloop	<kleur>red</kleur> <kleur>yellow</kleur>
$if	$if(conditie expression) DM$cript $endif	Een deel van DM$cript wordt uitgevoerd als aan voorwaarde 'conditie expression' wordt voldaan	**$if**($substr_before(//store catalogue/version,'.') = 2) <oke>true</oke> **$endif**	<oke>true</oke>
$index	$index	I.c.m. $loop te gebruiken: tellertje met het aantal iteratieslagen de loop/lus heeft gehad/nu mee bezig is	$loop($array(//fruit)) <teller>$value(**$index**)</teller> $endloop	<teller>1</teller> <teller>2</teller> <teller>3</teller>
$last	$last	I.c.m. $loop te gebruiken: het totaal aantal iteratieslagen van de loop/lus	$loop($array(//fruit)) <teller>$value(**$last**)</teller> $endloop	<teller>3</teller> <teller>3</teller> <teller>3</teller>
$loop	$loop(DM folder expressie) DM$script $endloop	Om een deel van de brondata repeterend te doorlopen en verwerken	**$loop**($array(//fruit)) <naam>$value(name)</naam> **$endloop**	<naam>apple</naam> <naam>banana</naam> <naam>strawberry</naam>
$not	$not(conditie expressie)	Logische NOT operator om een conditie expressie te gebruiken zodat niet aan de voorwaarde mag worden voldaan	$value(**$not**($substr_before(//store catalogue/version,'.') = 2))	false
$or	$or	Logische OR operator om in een conditie expressie te gebruiken zodat aan 1 van de 2 voorwaarden moet worden voldaan	$value($substr_before(//store catalogue/version,'.') = 3 **$or** $substr(//store catalogue/date, 7,4)='2024')	true
$orderby	$orderby(DM folder expression, datatype, sortingorder)	I.c.m. $loop te gebruiken, zodat de iteratieslagen van de loop/lus in een bepaalde volgorde gebeurt	$loop($array(//fruit))**$orderby**(color, txt, desc)**$orderby**(weight, num, asc)<naam>$value(name)</naam> $endloop	<naam>banana</naam> <naam>strawberry</naam> <naam>apple</naam>

Categorie: Searching

$altfollow				
$allpreced				
$anc				
$array	$array(DM folder expressie)	Om, typisch in geval van JSON, een array van waarden te kunnen doorlopen/verwerken	$value(**$array**(//fruit)[2]/name)	banana
$child				
$desc				
$endnamespace				
$follow				
$getattr				
$namespace				
$parent				
$preced				
$value	$value(DM folderexpressie)	Opvragen van waarde uit het veld aangegeven met de DM folderexpressie uit de brondata	**$value**(//store catalogue/date)	15-3-2024

Categorie: Text

$apos	$apos	Om in speciale/uitzonderlijke gevallen waar het niet lukt de apostroph ' geforceerd in de brondata op te nemen	$value($strcat('You',**$apos**,'re$spwelcome')	You're welcome
$cr	$cr	Om in speciale/uitzonderlijke gevallen waar het niet lukt de carriage return geforceerd in de brondata op te nemen		
$findstr	$findstr(string, string)	Om te controleren of in een tekst 'string' een andere tekst 'string' voorkomt	$value(**$findstr**(//store catalogue/date, '2024'))	true
$lf	$lf	Om in speciale/uitzonderlijke gevallen waar het niet lukt de linefeed geforceerd in de brondata op te nemen		
$lower	$lower(string)	De tekst 'string' wordt in kleine letters omgezet	$value(**$lower**('TEST'))	test
$lpad	$lpad(string, string, number)	Om een tekst 'string' aan de linkerzijde aan te vullen met een andere tekst totdat totaal de lengte 'number' is bereikt. Let op: lpad werkt niet icm $value	**$lpad**(//store catalogue/version,'000',5)	002.3
$quot	$quot	Om in speciale/uitzonderlijke gevallen waar het niet lukt de quote " geforceerd in de brondata op te nemen		
$replace	$replace(string, char, char)	Om in een tekst 'string' alle tekens 'char' aan te passen in een ander teken 'char'	$value(**$replace**(//store catalogue/date,'-','/'))	15/03/2024
$rpad	$rpad(string, string, number)	Om een tekst 'string' aan de rechterzijde aan te vullen met een andere tekst totdat totaal de lengte 'number' is bereikt. Let op: lpad werkt niet icm $value	**$rpad**(//store catalogue/version,'000',5)	2.300
$sp	$sp	Om in speciale/uitzonderlijke gevallen waar het niet lukt de spatie geforceerd in de brondata op te nemen	$value($strcat('Versie:**$sp**', //store catalogue/version, '**$sp**Datum:**$sp** ', //store catalogue/date))	Versie: 2.3 Datum: 15-03-2024
$strcat	$strcat(string, string, string, string,string)	Om meerdere teksten samen te voegen	$value(**$strcat**('Versie:$sp', //store catalogue/version, '$spDatum:$sp ', //store catalogue/date))	Versie: 2.3 Datum: 15-03-2024
$strlen	$strlen(string)	Geeft als resultaat de lengte (aantal tekens) van de tekst 'string'	$value(**$strlen**(//store catalogue/date))	10
$substr	$substr(string, number, number)	Uitlezen van een gedeelte van een tekst 'string' vanaf positie 'number' ter lengte van 'number'	$value(**$substr**(//store catalogue/date, 7,4)	2024
$substr_after	$substr_after(string, string)	Uitlezen van een gedeelte van een tekst 'string' vanaf de tekst 'string'	$value(**$substr_after**(//store catalogue/date, '-'))	03-2024
$substr_before	$substr_before(string, string)	Uitlezen van een gedeelte van een tekst 'string' tot aan de tekst 'string'	$value(**$substr_before**(//store catalogue/date, '-'))	15
$trim	$trim(string)	Om de spaties voor en achter een tekst 'string' te verwijderen	$value(**$trim**(' test '))	test
$upper	$upper(string)	De tekst 'string' wordt in hoofdletters omgezet	$value(**$upper**('test'))	TEST

Voorbeelden van aanroep van DataMixer API/webservice

In dit hoofdstuk wordt tot slot een samenvatting gegevens hoe de API van DataMixer (versie 1.0) kan worden aangeroepen, en worden voorbeelden (samples) in cUrl, Python en Javascript gegeven.

Je kunt de DataMixer-webservice aanroepen met een HTTP POST-request. Hier is hoe je dat doet met jouw XML als brondata en het DMScript om het naar JSON te converteren.

HTTP Request Structuur

📌 **Endpoint:**

https://datamixer.nl/api/1.0/?apikey=freeversion

📌 **HTTP Methode:**

POST

📌 **Headers:**

```
{
   "Content-Type": " text/plain"
}
```

📌 **Body (JSON input met XML-brondata en DMScript):**

```
{
  "DM": {
    "data": [
      {
        "content":
"<Memberlist><Member><Name>Peter</Name><Age>45</Age><Gender>m</Gender></Memb
er><Member><Name>Patrick</Name><Age>14</Age><Gender>m</Gender></Member><Memb
er><Name>Susan</Name><Age>32</Age><Gender>f</Gender></Member><Member><Name>
John</Name><Age>21</Age><Gender>m</Gender></Member><Member><Name>Jane</Nam
e><Age>16</Age><Gender>f</Gender></Member><Member><Name>Jessica</Name><Age>28
</Age><Gender>f</Gender></Member></Memberlist>"
      }
    ],
```

```
   "dm$": "$start('DataMixer') { \"MEMBERS\": [ $loop(//Memberlist/Member)
{\"NAME\": \"$value($upper(Name))\"}$if($index != $last),$endif $endloop ] }
$endstart"

  }

}
```

Voorbeeld met cURL

Je kunt dit uitvoeren in een terminal of command prompt:

```
curl -X POST "https://datamixer.nl/api/1.0/?apikey=freeversion" \

    -H "Content-Type: text/plain" \

    -d '{

  "DM": {

    "data": [

     {

       "content":
"<Memberlist><Member><Name>Peter</Name><Age>45</Age><Gender>m</Gender></Memb
er><Member><Name>Patrick</Name><Age>14</Age><Gender>m</Gender></Member><Memb
er><Name>Susan</Name><Age>32</Age><Gender>f</Gender></Member><Member><Name>
John</Name><Age>21</Age><Gender>m</Gender></Member><Member><Name>Jane</Nam
e><Age>16</Age><Gender>f</Gender></Member><Member><Name>Jessica</Name><Age>28
</Age><Gender>f</Gender></Member></Memberlist>"

     }

    ],

   "dm$": "$start('DataMixer') { \"MEMBERS\": [ $loop(//Memberlist/Member)
{\"NAME\": \"$value($upper(Name))\"}$if($index != $last),$endif $endloop ] }
$endstart"

  }

}'
```

Voorbeeld met Python (requests-bibliotheek)

```
import requests

import json


url = "https://datamixer.nl/api/1.0/?apikey=freeversion"

headers = {"Content-Type": " text/plain"}

payload = {

   "DM": {

      "data": [

         {

            "content":
"<Memberlist><Member><Name>Peter</Name><Age>45</Age><Gender>m</Gender></Memb
er><Member><Name>Patrick</Name><Age>14</Age><Gender>m</Gender></Member><Memb
er><Name>Susan</Name><Age>32</Age><Gender>f</Gender></Member><Member><Name>
John</Name><Age>21</Age><Gender>m</Gender></Member><Member><Name>Jane</Nam
e><Age>16</Age><Gender>f</Gender></Member><Member><Name>Jessica</Name><Age>28
</Age><Gender>f</Gender></Member></Memberlist>"

         }

      ],

      "dm$": "$start('DataMixer') { \"MEMBERS\": [ $loop(//Memberlist/Member)
{\"NAME\": \"$value($upper(Name))\"}$if($index != $last),$endif $endloop ] }
$endstart"

   }

}

response = requests.post(url, headers=headers, json=payload)

print(response.text)  # Geeft de geconverteerde JSON terug
```

JavaScript (Fetch API) Voorbeeld

```
const url = "https://datamixer.nl/api/1.0/?apikey=freeversion";

const headers = { "Content-Type": " text/plain" };

const payload = {

  "DM": {

    "data": [

      {

        "content":
"<Memberlist><Member><Name>Peter</Name><Age>45</Age><Gender>m</Gender></Memb
er><Member><Name>Patrick</Name><Age>14</Age><Gender>m</Gender></Member><Memb
er><Name>Susan</Name><Age>32</Age><Gender>f</Gender></Member><Member><Name>
John</Name><Age>21</Age><Gender>m</Gender></Member><Member><Name>Jane</Nam
e><Age>16</Age><Gender>f</Gender></Member><Member><Name>Jessica</Name><Age>28
</Age><Gender>f</Gender></Member></Memberlist>"

      }

    ],

    "dm$": "$start('DataMixer') { \"MEMBERS\": [ $loop(//Memberlist/Member)
{\"NAME\": \"$value($upper(Name))\"}$if($index != $last),$endif $endloop ] }
$endstart"

  }

};

async function callDataMixer() {

  try {

    const response = await fetch(url, {

      method: "POST",

      headers: headers,

      body: JSON.stringify(payload)

    });
```

```
    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }
    const data = await response.text();
    console.log("DataMixer Output:", data);
  } catch (error) {
    console.error("Error calling DataMixer:", error);
  }
}
callDataMixer();
```

Disclaimer

Versie 1.0 van Data Mixer is een "free-to-use" editie, waarbij de volgende punten van toepassing zijn:

- Deze dienst wordt kosteloos aangeboden,
- Er bestaat geen enkele resultaatverplichting op deze dienst,
- Er geldt een fair-use policy t.a.v. het gebruik van deze dienst,
- Deze dienst biedt geen enkele garantie op performance, het is daarom mogelijk dat de responsesnelheid niet voldoet uw verwachtingen,
- Deze dienst kan zonder kennisgeving tijdelijk of voor langere tijd buiten werking zijn, b.v. als gevolg van onderhoudswerkzaamheden,
- De door uw aangeboden data wordt op geen enkele wijze gelogd en/of bewaard,
- Er wordt geen support geboden op deze dienst,
- U bent in het kader van de AVG wetgeving zelf verantwoordelijk voor de door u aangeboden databronnen en gegevens,
- U bent zelf verantwoordelijk of en hoe de verwerking van de door u aangeboden databronnen en gegevens plaatsvindt.

De free-to-use editie stelt (voor iedereen) een algemene api-key beschikbaar, waarbij met in achtneming van de vorige punten ook de volgende punten van toepassing zijn:

- De algemene apikey is: freeversion,
- Hiermee wordt u als gebruiker in staat gesteld om de webservice (api) van DataMixer aan te roepen / te gebruiken,
- Gebruik van deze algemene apikey kan bij (te) grote drukte extra wachttijd / responsetijd betekenen.