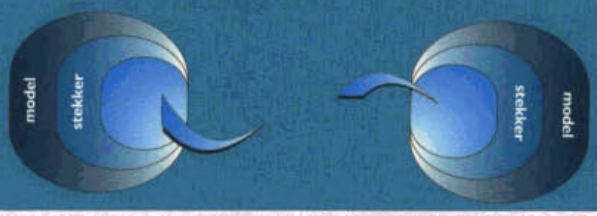


stowa

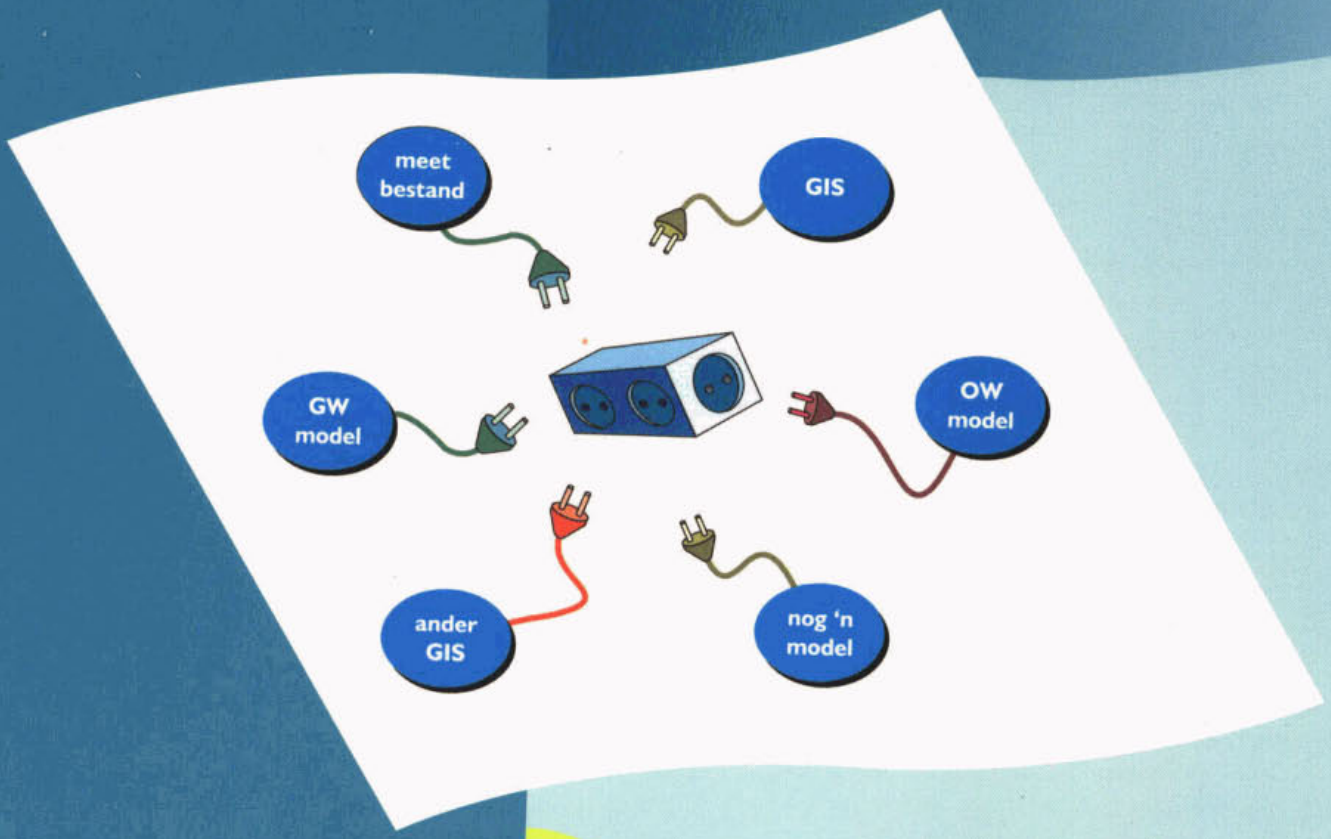
Stichting Toegepast Onderzoek Waterbeheer

1999-28_stekkerdoos-water-versie-3

STEKKERDOOS WATER

versie 3.0

- *Gebruikershandleiding*
- *Functioneel ontwerp*
- *Technisch ontwerp*



99 28

stowa

Stichting Toegepast Onderzoek Waterbeheer

STEKKERDOOS WATER
versie 3.0

- Gebruikshandleiding

- Functioneel ontwerp

- Technisch ontwerp

99 28

Arthur van Schendelstraat 816
Postbus 8090, 3503 RB Utrecht
Telefoon 030 232 11 99
Telefax 030 232 17 66
E-Mail stowa@stowa.nl

Internet www.waterland.net/stowa

ISBN 90.5773.078.2

TEN GELEIDE

In juli 1998 is de Stekkerdoos Water versie 2, gefinancierd door IPO, RIZA, Unie van Waterschappen en STOWA, opgeleverd. De Stekkerdoos Water is een systeem waarmee gegevens die volgens de Gegevenstandaard Water (ADVENTUS) of conform de CIW-gegevensstandaard zijn geclassificeerd eenvoudig kunnen worden geschreven en gelezen naar een uitwisselingsbestand. Dit uitwisselingsbestand is bedoeld voor de overdracht van gegevens tussen organisaties, voor het uitwisselen van waardereeksen (zoals meetreeksen en rekenresultaten) en gegevensuitwisseling tussen applicaties.
Voor het schrijven en lezen van deze bestanden wordt enerzijds een bibliotheek beschikbaar gesteld met high-level functies en daarnaast een set van hulpmiddelen die het de stekkerbouwer gemakkelijk maakt om de gegevens op de juiste manier te declareren en te behandelen in de stekkers. De gebruikers van de Stekkerdoos Water zijn de programmeurs die stekkers bouwen.
De in eerste instantie gebruikte oplossingmethodiek is zeer geschikt gebleken voor off-line koppeling. Nu voor on-line koppeling de performance eisen zo belangrijk geworden zijn dient een ander concept te worden gehanteerd daar het huidige concept volledig 'uitgemolken' is. Bij toepassing van dit nieuwe concept worden tevens vijf extra verbeterpunten, waaronder performance, integraal aangepakt en opgelost. Deze werkzaamheden hebben geresulteerd in de Stekkerdoos Water versie 3.0.
De werkzaamheden zijn uitgevoerd door WL|Delft Hydraulics, met als projectleider de heer J.F.M. Overmars en projectmedewerker dhr. C. van der Schelde en zijn namens de STOWA begeleid door ir. L.R. Wentholt onder verantwoording van de STOWA stuurgroep IT.

Utrecht, december 1999

De directeur van de STOWA

Ir. J.M.J. Leenen

Stekkerdoos Water (versie 3.0)

Gebruikershandleiding

Inhoud

1 Inleiding 1-1

1.1 Systeemnaam en titel 1-1

1.2 Achtergrond en doelstelling 1-1

1.3 Toepassingsgebied 1-2

1.4 Functionele beschrijving 1-3

2 Datamodellering 2-1

2.1 Begrippen 2-1

2.2 Waardenreeksen in de Stekkerdoos 2-2

3 Systeembeschrijving 3-1

3.1 Adventus gegevensmodel 3-2

3.2 Stuurbestandgenerator 3-2

3.3 Adventus stuurbestand 3-3

3.4 Bilateraal stuurbestand 3-3

3.5 Generator voor het uitwisselingsformaat 3-4

3.6 Het uitwisselingsformaat 3-5

3.7 Stekkerdoosfuncties 3-6

3.7.1 Algemeen 3-6
3.7.2 Performance 3-6
3.7.3 Functies 3-7

3.8 Applicatie/stekker 3-8

3.8.1 Algemene opmerkingen 3-9

4 DLL 4-1

4.1 Algemeen 4-1

4.2 Digital Visual Fortran 4-1

4.3 Microsoft C/C++ en Visual Basic 4-1
4.3.1 Inleiding 4-1
4.3.2 Fortran CHARACTER 4-2
4.3.3 Visual Basic en karakters 4-2
4.3.4 C/C++ en karakters 4-3

5 Installatie 5-1

5.1 Systeemeisen 5-1
5.2 Platforms en portabiliteit 5-1
5.3 Installatie-instructie 5-1
5.4 Programmeur instructies 5-2
5.5 Overzicht files op diskette 5-2
5.5.1 Bibliotheek bestanden 5-2
5.5.2 Documentatie 5-2
5.5.3 C-interface 5-2
5.5.4 Visual Basic 5-3
5.5.5 Voorbeelden 5-3
5.5.6 Executables en SDW-scripts 5-3
5.5.7 Documentatie 5-3
5.5.8 Stuurbestand Adventus (GW96) en voorbeeld bilateraal bestand 5-4

6 Beschrijving van de functies 6-1

6.1 Overzicht en algemene opmerkingen 6-1
6.2 Foutmeldingen 6-2
6.3 Detail beschrijving van de functies 6-3
6.3.1 SWALLOC 6-3
6.3.2 SETVAL 6-4
6.3.3 OPNUWB 6-5
6.3.4 CLSUWB 6-6
6.3.5 CRETAB 6-7
6.3.6 WRVLD 6-8
6.3.7 WRREC 6-9
6.3.8 RDREC 6-11
6.3.9 RDVLD 6-13
6.3.10 WRKOL 6-14
6.3.11 RDKOL 6-15
6.3.12 CREWRD 6-17
6.3.13 WRWRD 6-18
6.3.14 RDWRD 6-19
6.3.15 INQUWB 6-21

6.3.16 INQTBL 6-22
6.3.17 INQWRD 6-24

7 Voorbeelden 7-1

7.1 Schrijven van een tabel 7-1

7.2 Lezen van een tabel 7-3

7.3 Schijven van waardenreeksen 7-6

7.4 Lezen van waardenreeksen 7-8

7.5 Opvraagfuncties 7-11

8 Beschrijving stuurbestanden 8-1

8.1 GW-(Adventus) classificatie 8-1

8.1.1 Deel 1: Administratieve gegevens 8-1
8.1.2 Deel 2: Gegevenselementen 8-1
8.1.3 Deel 3: Entiteittypen 8-2

8.2 Bilaterale afspraken 8-3

8.2.1 Algemeen 8-3
8.2.2 Deel 1: Administratieve gegevens 8-3
8.2.3 Deel 2: Gegevenselementen 8-3
8.2.4 Deel 3: Entiteittypen 8-4

8.3 Voorbeeld GW96 stuurbestand 8-5

8.4 Voorbeeld bilateraal stuurbestand 8-8

9 Overzicht Nefis foutmeldingen 9-1

1 Inleiding

1.1 Systeemnaam en titel

De **Stekkerdoos Water** is een systeem waarmee gegevens, die volgens de Gegevensstandaard Water zijn geclassificeerd, eenvoudig kunnen worden geschreven naar en gelezen vanuit een uitwisselingsbestand. Dit uitwisselingsbestand is bedoeld voor de overdracht van gegevens tussen applicatieprogramma's onderling, maar ook voor de overdracht van gegevens tussen organisaties.

Daarnaast biedt de Stekkerdoos Water mogelijkheden voor het uitwisselen van waardenreeksen zoals meetreeksen en rekenresultaten via hetzelfde uitwisselingsbestand.

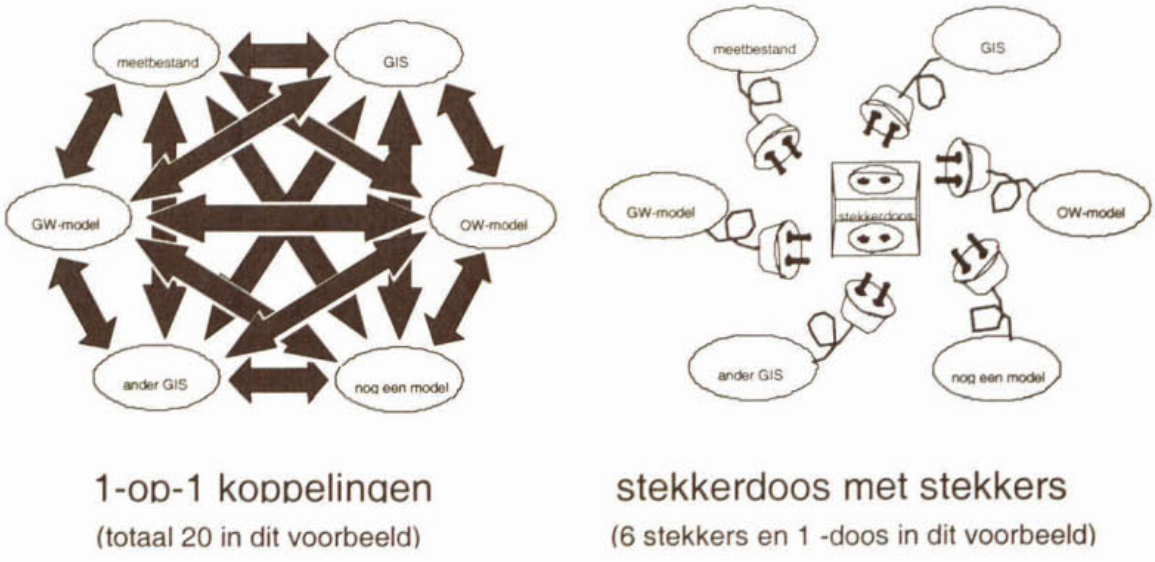
De Stekkerdoos Water is een DLL met functies, die gebruikt kan worden voor het realiseren van zogenoemde *Stekkers*. De **gebruikers** van de Stekkerdoos Water zijn dus de programmeurs die stekkers bouwen. Dit document is de handleiding voor deze groep van gebruikers.

Gebruikerswensen en -eisen hebben geleid tot een re-design van de Stekkerdoos Water en deze is nu bekend als versie 3.0. Dit hernieuwde ontwerp heeft tot gevolg gehad dat functies en uitwisselingsbestand nogal veranderd zijn. De consequentie hiervan is dat uitwisselingsbestanden die met eerdere versies zijn gemaakt, niet gelezen kunnen worden door functies uit versie 3.0. Dit geldt ook andersom waarbij een uitwisselingsbestand gemaakt met functies van versie 3.0 niet kan worden gelezen door eerdere versies van de Stekkerdoos Water.

1.2 Achtergrond en doelstelling

Er is een groeiende behoefte aan gegevensuitwisseling tussen informatiesystemen van dezelfde of van verschillende organisaties. Een randvoorwaarde voor succesvolle uitwisseling en hergebruik is dat de gegevens op een eenduidige manier geclassificeerd zijn. Hiertoe is door de Unie van Waterschappen een Gegevensstandaard Water opgesteld. De Stekkerdoos Water biedt functies voor het uitwisselen van gegevens die volgens dit classificatiemodel zijn gemodelleerd.

Voor de uitwisseling van gegevens tussen bestaande applicaties is veelal een conversieslag nodig. Door nu alle conversies te laten plaatsvinden via een standaard tussenformaat, wordt bewerkstelligd dat voor elke applicatie slechts één conversie behoeft te worden gebouwd in plaats van een conversie per combinatie van applicaties. Dit wordt grafisch in beeld gebracht in Figuur 1.

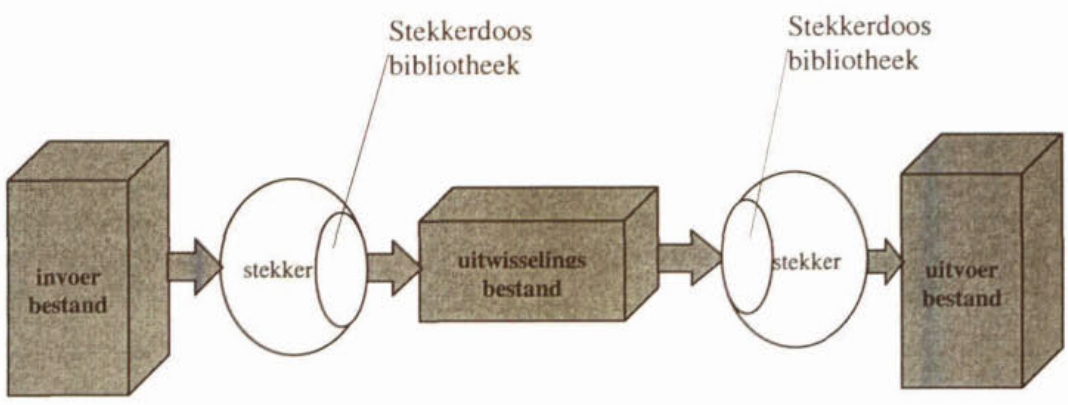


Figuur 1: Functie van de Stekkerdoos Water

Het is mogelijk van het Adventus datamodel af te wijken door om afwijkingen en aanvullingen hierop vast te leggen in een zogenaamd *bilateraal stuurbestand*. Hiermee is het mogelijk om organisatie-specifieke classificaties uit te wisselen in aanvulling op de Adventus classificaties.

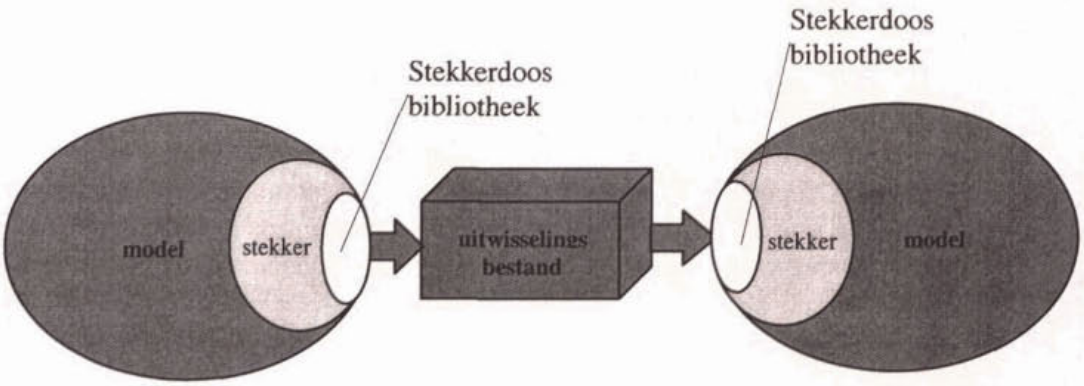
1.3 Toepassingsgebied

De Stekkerdoos Water is geschikt voor gebruik in programma's die gegevensbestanden omzetten naar uitwisselingsbestanden, zogenaamde stekkers, zoals aangegeven in Figuur 2.



Figuur 2: Omzetting van bestanden naar uitwisselingsbestanden

Het is ook mogelijk de Stekkers te integreren in andere applicaties zoals waterbewegingsmodellen, grondwatermodellen, GIS-applicaties en dergelijke. Figuur 3 geeft hiervan een voorbeeld.



Figuur 3: On-line gegevensoverdracht van en naar het uitwisselingsbestand

In alle gevallen zal er een vertaling moeten plaats vinden van de classificatie zoals gebruikt in de modellen of de in-/uitvoerbestanden naar de Adventus classificatie (en mogelijk de aanvullende bilateraal afgesproken classificatie). Deze vertaling (ook wel *mapping* genoemd) moet per model of per in-/uitvoerbestand worden geprogrammeerd.
De code waarin deze vertaling wordt verzorgd wordt een *Stekker* genoemd. Bij het gebruik van de Stekkerdoos Water binnen een model of applicatie is de stekker dus geïntegreerd met het model.
Bij omzetting van databestanden naar uitwisselingsbestanden is de stekker een specifiek zelfstandig programma.

De Stekkerdoos Water biedt naast de mogelijkheid om gegevens gemodelleerd volgens een relationeel model uit te wisselen, ook de mogelijkheid om meetreeksen en tijdreeksen en resultaten van 1d- , 2d- en 3dimensionel rekenmodellen uit te wisselen, de zogenaamde *waardenreeksen.*

1.4 Functionele beschrijving

De Stekkerdoos Water heeft functies om gegevens in de vorm van tabellen en waardenreeksen naar/van een uitwisselingsbestand te schrijven/lezen. Voor tabellen kan gebruik worden gemaakt van functies om velden en records van een tabel te schrijven en te lezen. Ook is het mogelijk om kolommen van een tabel te schrijven en te lezen. Voor waardenreeksen zijn er functies voor lezen van en schrijven naar het uitwisselingsbestand. In hoofdstuk 2 wordt nader ingegaan op de begrippen waardenreeksen en de tabellen.

Voor een goede werking van de genoemde functies uit punt 1 hierboven is nodig dat tabellen en waardenreeksen zijn beschreven. Hiervoor wordt het zogenoemde *uitwisselingsformaat* gebruikt. Dit is een apart bestand dat door de Stekkerdoosfuncties wordt gebruikt.

Het bestand met het uitwisselingsformaat wordt gemaakt door een apart programma CREUWF. Dit programma gebruikt zogenaamde stuurbestanden als invoer. Er is een stuurbestand gebaseerd op Adventus en een zogenoemd bilateraal stuurbestand dat een aanvulling of uitbreiding is op het Adventus stuurbestand. Hoofdstuk 3 geeft een beschrijving van de stuurbestanden. In hoofdstuk 3.5 wordt verder ingegaan op CREUWF en de stuurbestanden.

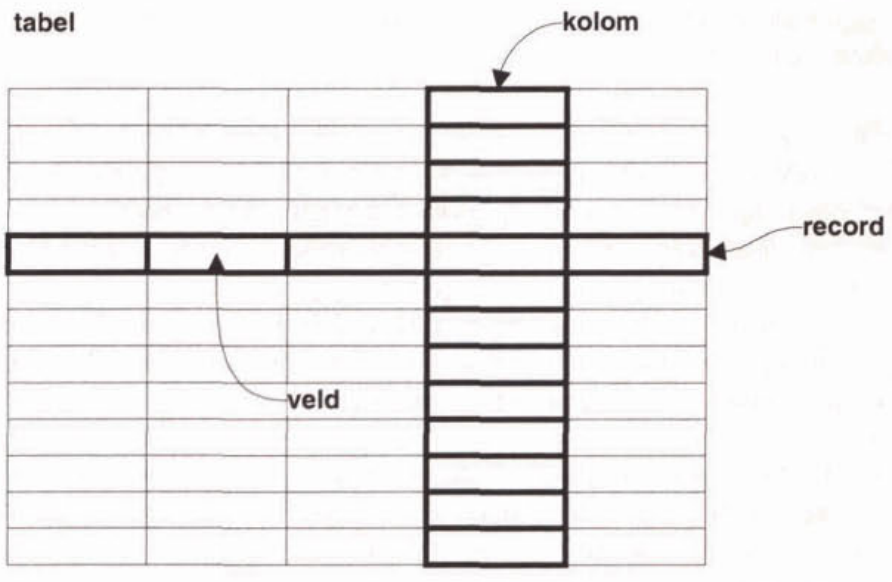
Het Adventus stuurbestand wordt meegeleverd met de Stekkerdoos Water, alsmede een voorbeeld van een bilateraal stuurbestand.
Een bilateraal stuurbestand wordt handmatig aangemaakt. Het Adventus stuurbestand is gemaakt via een SDW-script uit de originele Adventus beschrijving (classificatie) welke gemaakt is in SDW. SDW staat voor System Development Workbench en is een product van CapGemini. In hoofdstuk 3.3 wordt nader ingegaan op het genereren van het Adventus stuurbestand.

2 Datamodellering

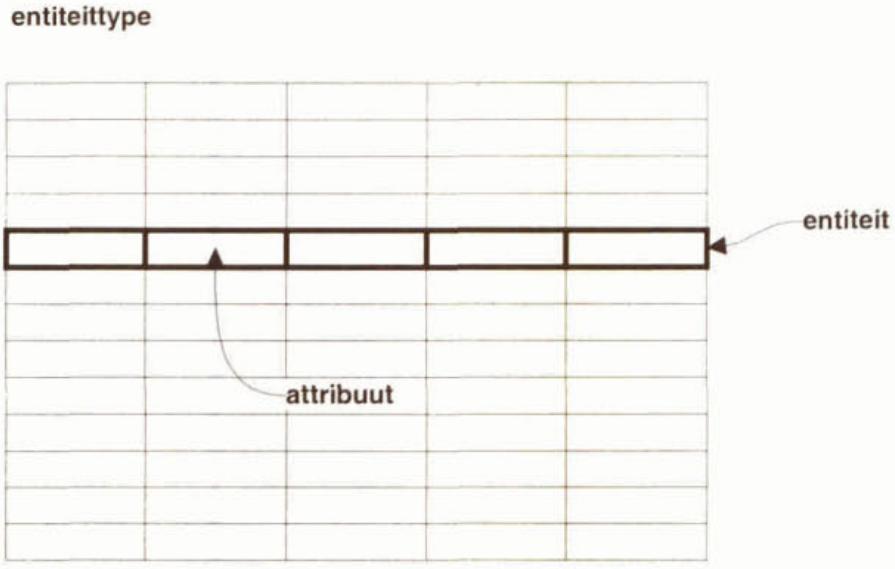
Bij de toepassing van de Stekkerdoos Water wordt ervan uitgegaan dat de gebruiker/programmeur bekend is met datamodellering en met het Adventusstelsel. Onderstaand wordt ingegaan op de begrippen uit datamodellering en de gebruikte terminologie binnen de Stekkerdoos Water. Daarna wordt het begrip *waardenreeksen*, zoals dat wordt gehanteerd binnen de Stekkerdoos, uitvoerig uitgelegd.

2.1 Begrippen

Binnen de datamodellering wordt gebruik gemaakt van de begrippen: entiteittype, entiteit en attribuut.
Een entiteittype is een collectie, of een verzameling, van entiteiten.
Een entiteit wordt beschreven door een aantal attributen (kenmerken).



Voor de fysieke opslag worden deze begrippen vertaald naar tabellen, waarbij een tabel opgebouwd is uit meerdere kolommen, records en velden.
Een record (=entiteit) is één voorkomen binnen de gehele tabel (=entiteittype).
Een record (=entiteit) zelf is opgebouwd uit meerdere velden (=attributen).



In dit ontwerp zal niet meer gesproken worden in termen van entiteittype, entiteit en attribuut, maar in termen van *tabel*, *record*, *veld* en *kolom* als het betrekking heeft op het Adventus gegevensmodel.

Bij het ontwerp van de eerste versie van Stekkerdoos Water is onderkend dat ook reeksenwaarden, zoals tijdreeksen, uitgewisseld moeten kunnen worden. Destijds is de term *waardenreeks* ingevoerd voor een bij elkaar behorende verzameling waarden, opgeslagen in arrays. Een andere tijdreeks is een nieuwe verzameling waarden, van het type tijdreeks en met een unieke naam.
Daarmee is een Stekkerdoos waardenreeks iets anders dan het entiteittype 'waardereeks' uit het Adventus gegevensmodel! In Adventus worden bij alle (meet)waarden opgeslagen in de tabel 'Meetwaarde' en is er een relatie naar een tijdreeks.

De Stekkerdoos-waardenreeks wordt in paragraaf 3.3.3. uitvoerig uitgelegd. Ook daar komen we tot begrippen entiteittype en attributen. Daarmee zouden we ook voor waardenreeksen kunnen spreken van tabellen en velden.
Echter, het enigszins bijzondere karakter van waardenreeks als bij elkaar behorende arrays van waarden, en ook de bekendheid uit vorige versies van de Stekkerdoos hebben er toe geleid om voor waardenreeksen als entiteittype de termen waardenreeks en attribuut te gebruiken in plaats van tabel en veld.
Waardenreeksen worden onder een unieke naam op het uitwisselingsbestand opgeslagen.

2.2 Waardenreeksen in de Stekkerdoos

Uitleg van het begrip waardenreeks
In het functioneel ontwerp van Stekkerdoos versie 1.0 is het begrip *waardenreeks* reeds geïntroduceerd. De daar gegeven definitie zegt: "waardenreeksen zijn (lange) reeksen van getallen met een vaste structuur."

In dit document willen we het begrip *waardenreeks* opnieuw beschrijven, een verband leggen met entiteittypen en van daaruit aangeven hoe waardenreeksen op het bilaterale

stuurbestand worden gedefinieerd. Daarbij wordt aangegeven hoe de programmeur of gebruiker met waardenreeksen kan omgaan.

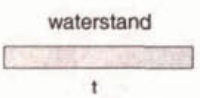
Als eerste kunnen we zeggen dat waardenreeksen attributen bezitten (die de kenmerken weergeven). De attributen van de waardenreeks hebben een zekere afmeting, terwijl de waardenreeks zelf een lengte heeft.
Voorbeelden waardenreeksen zijn:

- Tijdreeks: heeft een lengte en bestaat uit de attributen Datum, Tijd, Waarde;
- Polygoon: heeft een lengte en bestaat uit de attributen X-coördinaat, Y-coördinaat;
- Grid: heeft een 'lengte' bestaat uit het attribuut Coördinaten, met de afmeting m*n;
- Rooster: heeft een 'lengte' en bestaat uit de attributen X-coördinaat en Y-coördinaat, ieder met de afmeting n*m;
- 3Dblok: heeft een 'lengte' en het attribuut Parameters met de afmeting m*n*t

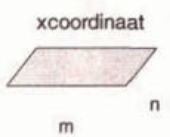
Wel is het zo dat ieder attribuut van een waardenreeks exact dezelfde afmeting heeft. Aan het einde van deze paragraaf wordt verder ingegaan op de begrippen lengte voor waardenreeks en afmeting voor attribuut.

Een aantal voorbeelden van waardenreeksen (met één attribuut) zijn:

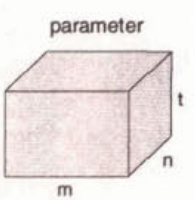
- een tijdreeks met waterstanden voor een bepaalde locatie



- een matrix met de x-coördinaten van een 2-dimensionaal rekenrooster,



- een 3d matrix met de parameterwaarden (bijv. waterstanden) van een2-dimensionaal rekenrooster in de tijd,



Meer algemeen kan worden gezegd: een n-dimensionale matrix met waarden, waarbij n loopt van 1..5. De applicatie en niet de stekkerdoos kent de betekenis van de structuur!

Formele beschrijving van waardenreeks als entiteittype met attributen
Kijken we nu op een formeel of logisch niveau naar waardenreeksen, dan kan ook gezegd worden dat de waardenreeksen entiteittypen zijn en dat de kenmerken de attributen zijn. Er zijn dan entiteittypen als bijvoorbeeld Tijdreeks, Polygoon, Rooster, Grid met als attributen matrices of verzamelingen met waarden. Dus hier is een attribuut meer dan één afzonderlijke waarde (zoals bij 'gewone' entiteiten). Op deze wijze sluit dit alles op logisch niveau aan bij hetgeen in het Adventus model is toegepast en daarmee ook bij de opzet van het stuurbestand.

Bij 'gewone' entiteittypen ontstaan nieuwe voorkomens via een sleutelwaarde en een nieuw record in de tabel. De Stekkerdoos Water zorgt voor opslag in het uitwisselingsbestand via

functies als *creëer tabel* van een entiteittype, *vullen velden*, *vullen sleutelvelden* en *schrijf record*. Natuurlijk zijn er ook de leesfuncties.
Bij de waardenreeksen ontstaan nieuwe voorkomens via een unieke naam (=sleutel) en het aangeven van het entiteittype waarvan deze is afgeleid. De Stekkerdoos Water zorgt voor opslag in het uitwisselingsbestand via functies als *creëer waardenreeks* en *schrijf (waarden)reeks*.

De waardenreeksen hebben als attributen *verzamelingen (matrices)* met waarden en niet *één afzonderlijke* waarde, zoals bij de gewone entiteittypen. Deze attributen zijn allemaal wel van éénzelfde type (1d, 2d, 3d... n-dimensionale matrix), zoals opgegeven bij de beschrijving van de waardenreeks.
De attributen (=velden) kunnen nu niet meer gevuld worden met een functie als *vullen veld* of *ophalen veld* en vervolgens *lees record* of *schrijf record*. Omdat de attributen van waardenreeksen veelal een forse hoeveelheid waarden zijn, wordt een dergelijk attribuut in één keer naar file geschreven of ervan gelezen. Deze functies zouden een naam als *schrijf waardenreeks-attribuut* kunnen hebben en idem voor het lezen. Om pragmatische reden is gekozen voor de naam *schrijf reeks* of *lees reeks* omdat attributen al snel reeksen worden genoemd (denk aan het veel gebruikte voorbeeld Tijdreeks)

Het schrijven en lezen van waardenreeksen
Hierbij kunnen de volgende stappen worden onderscheiden:

1. Als eerste is er de definitie van de waardenreeks als entiteittype met een naam en met attributen die een naam en afmeting hebben. Deze worden opgegeven in het zogenaamde bilaterale stuurbestand.
2. Creëer daarna via een functie als bijvoorbeeld *crewrd* een unieke waardenreeks met naam en van een bepaald entiteittype; bijvoorbeeld T1 van type Tijdreeks.
3. Schrijf dan de waarden (= alle attributen) van T1 naar het uitwisselingsbestand, dat wil zeggen: schrijf de afzonderlijke attributen Datum, Tijd, Waarde waaruit de tijdreeks bestaat.
4. Voor andere waardenreeksen, zoals bijvoorbeeld van het type Rooster en 3Dblok, geldt dezelfde procedure.

Relatie met Adventus
Adventus kent ook het entiteittype '*waardereeks*', zie ERD metingen.2 Deze entiteit heeft attributen voor het vastleggen van gegevens omtrent een waardenreeks. Het entiteittype kan bijvoorbeeld worden uitgebreid met attributen voor de unieke naam (sleutel) van een waardenreeks en zijn type, zoals hierboven beschreven. Daarmee wordt een relatie gelegd naar de specifieke waardenreeks zoals we die kennen in de Stekkerdoos.

Voor alle duidelijkheid: hierboven is het begrip waardenreeks beschreven zoals gebruikt binnen de Stekkerdoos Water. Er ontstaan entiteittypen als Tijdreeks, Polygoon, Rooster, Grid met als attributen verzameling matrices met waarden. Het betreft hier steeds de opslag van de waarden. Het entiteittype '*waardereeks*' uit Adventus geeft aanvullende informatie over waardenreeksen en een relatie naar de daadwerkelijke verzameling waarden.

Relaties tussen entiteiten en waardenreeksen

Een waardenreeks wordt geïdentificeerd door zijn unieke naam. Hiervoor zijn tabellen aanwezig binnen de Stekkerdoos Water. Een waardenreeks behoeft niet gekoppeld te zijn aan een andere entiteit. Anderzijds is het mogelijk dit wel te doen via bijvoorbeeld het Adventus entiteittype '*waardereeks*' (zie hierboven).
Het is ook mogelijk relaties te leggen tussen waardenreeksen en andere entiteittypen. Daarvoor moeten deze entiteittypen worden uitgebreid met nieuwe attributen, o.a. voor de naam van de waardenreeks.

Afmeting/lengte van waardenreeksen

Hierboven is reeds opgemerkt dat de attributen van waardenreeksen een vaste structuur hebben. Aan de andere kant is het gewenst (prettig) dat bij het creëren van een waardenreeks toch een afmeting (een lengte) kan worden opgegeven. Via het uitwerken van voorbeelden willen we het verband tussen *attribuutdimensie* (=afmeting van het attribuut) en de *lengte van een waardenreeks* aangeven.

In het eerder genoemde voorbeeld Tijdreeks hebben we te maken met een 1-dimensionale reeksen als attributen. Dit is ook te zien als een 1-dimensionale array.
We komen tot deze 1-dimensionale array door bij de definitie van entiteittype Tijdreeks bij de attributen de attribuutdimensie 1 op te geven (= 1 enkel element van de 1d array). Bij het creëren van een tijdreeks met bijv. naam T1 geven we de lengte van de waardenreeks op, bijvoorbeeld 102. De combinatie van attribuutdimensie 1 met lengte 102 geeft de 1d array (van 102) voor de attributen Datum, Tijd en Waarde van Tijdreeks T1.

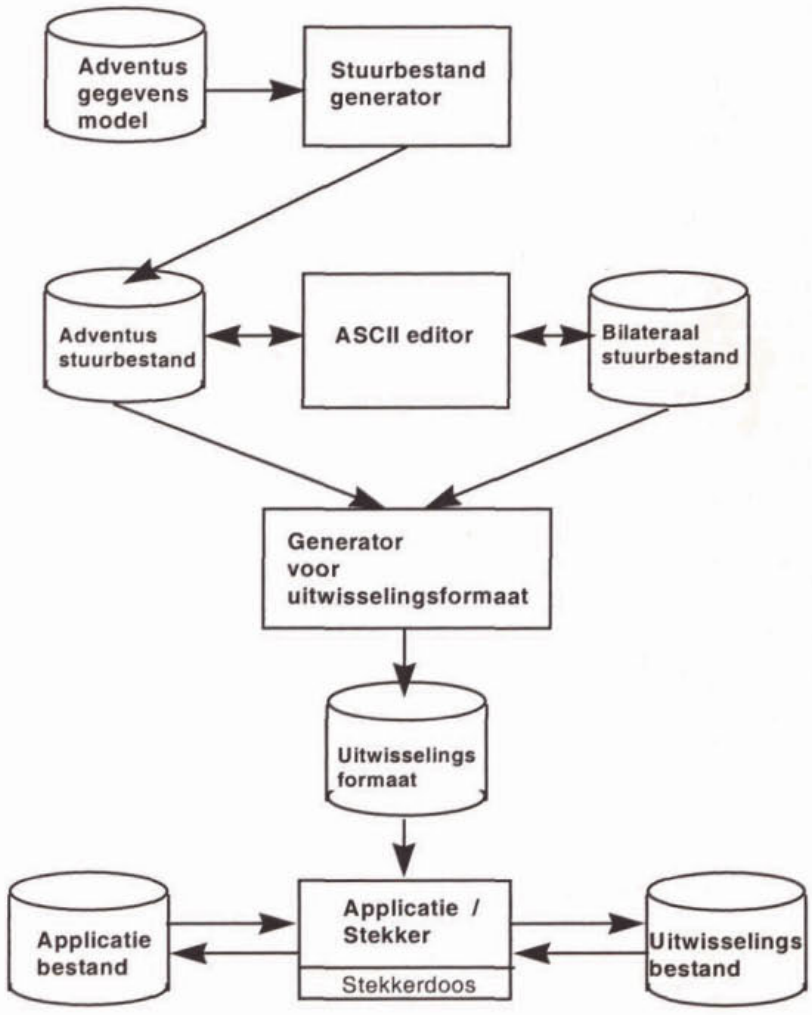
In het geval een entiteittype Punt met als attribuut "X-Y" voor de x, y coördinaten bevat de definitie van Punt het attribuut "X-Y" met attribuutdimensie 2. Hierin worden de x en y waarden opgeslagen. Bij de creatie van een waardenreeks met naam Punt-A krijgt deze de waardenreekslengte van bijvoorbeeld 145, om 145 punten weg te kunnen schrijven. Bij het schrijven naar het uitwisselingsbestand hebben we dus te maken met een 2d array van 2 * 145.

Voor de al eerder genoemde voorbeelden als Grid en Rooster geldt eenzelfde aanpak. Bij de definitie wordt als attribuutdimensie opgegeven bijvoorbeeld 100 * 275 voor het grid of rooster. Bij de creatie wordt dan een waardenreekslengte van 1 opgegeven. Daarmee krijgt de unieke waardenreeks met naam GRID-X van entiteittype GRID voor het attribuut "Coördinaten" een 3-dimensionale array van 100 * 275 * 1, die naar het uitwisselingsbestand geschreven wordt.
Zo kunnen met deze getalvoorbeelden voor waardenreeks R1 van entiteittype Rooster de twee attributen "X-coördinaat" en "Y-coördinaat" als twee afzonderlijke 3-d arrays van 100 * 275 * 1 worden weggeschreven.
Voor het genoemde voorbeeld 3Dblok geven we bij de definitie een vaste attribuutdimensie op (dat is het rekenrooster waarop de parameterwaarden zijn gebaseerd), zeg 112 * 245. Bij de creatie van een blok met parameterwaarden geven we de waardenreekslengte op van bijvoorbeeld 7. Hiermee worden dan in een 3-d matrix van 112 * 245 * 7, de 7 gewenste parameters (gebaseerd op het grid van 112 * 245) naar het uitwisselingsbestand geschreven.

NB
Bij het wegschrijven en teruglezen van arrays naar/uit het uitwisselingsbestand wordt de opslag van de waarden in de arrays beschouwd als een aaneengesloten geheugenblok. De kennis over de volgorde van de afzonderlijke waarden is aanwezig in de applicatieprogrammatuur.

3 Systeembeschrijving

Onderstaande figuur geeft een samenhang van het totale systeem Stekkerdoos Water



De volgende processen en bestanden zijn aanwezig:

- Adventus gegevensmodel
- de stuurbestandgenerator, CRESTBST
- Adventus stuurbestand of GW96 stuurbestand
- bilateraal stuurbestand
- generator voor het uitwisselingsformaat, het programma CREUWF
- het bestand met uitwisselingsformaat
- een applicatie met daarin opgenomen de Stekkerdoos functies (hier aangegeven als een lees- en een schrijfstekker)
- het uitwisselingsbestand (voor schrijven en lezen)

Deze processen en bestanden worden in de volgende paragrafen uitgewerkt.

3.1 Adventus gegevensmodel

Het Adventus gegevensmodel (Gegevensstandaard Water) is een logisch gegevensmodel met daarin een classificatie van aan water gerelateerde gegevens. Adventus is opgesteld door de Unie van Waterschappen. Adventus is daar te verkrijgen, maar wordt niet met de Stekkerdoos meegeleverd.
De Stekkerdoos Water is gebaseerd op de SDW® versie van deze gegevensstandaard. Dit houdt in dat voor het gebruik van deze standaard een versie van SDW® (module DM) beschikbaar moet zijn.
SDW staat voor System Development Workbench en is een product van CapGemini.

In Adventus zijn waardereeksen opgenomen. Deze modellering van waardereeksen is onvoldoende voor die situaties waar het gaat om (zeer) grote aantallen waarden en (zeer) veel waardereeksen te modelleren. In de toekomst zullen waarschijnlijk veranderingen rondom waardereeksen plaatsvinden. Zolang dit nog niet het geval is werkt de Stekkerdoos met een eigen definitie van waardenreeksen, zie hiervoor hoofdstuk 2.

In het SDW model van Adventus zijn entiteittypen geclusterd in superentiteittypen. Deze superentiteittypen hebben attributen die voor alle subentiteittypen van toepassing zijn. In de omzetting van het Adventus gegevensmodel tot stuurbestand worden de attributen van de superentiteittypen gekopieerd naar de subentiteittypen. Verder worden ook de superentiteittypen als aparte entiteittypen gedefinieerd. Op deze wijze is het mogelijk om informatie die bij een groep van entiteittypen behoort uit te wisselen, zonder de entiteittypen afzonderlijk te behandelen.

De Stekkerdoos Water neemt alle attributen van de entiteittypen mee dus ook de sleutelvelden. Meerdere sleutelvelden kunnen tezamen worden gebruikt als bijvoorbeeld een primary key. Adressering in de uitwisselingsfile vindt plaats aan de hand van deze (primaire) sleutels.

De Stekkerdoos Water biedt geen faciliteit om te achterhalen of een bepaald attribuut een foreign key is. Foreign keys worden op dezelfde manier meegenomen als andere attributen en kunnen op dezelfde manier worden gebruikt.

3.2 Stuurbestandgenerator

De stuurbestandgenerator is een SDW script met de naam CRESTBST.scr.

Wanneer gebruiken

Gebruik dit programma alleen als vanuit SDW® een nieuw Adventus-stuurbestand moet worden gegenereerd. Zodra een stuurbestand beschikbaar is hoeft dit programma niet meer te worden gebruikt, totdat er een nieuwe versie van Adventus beschikbaar komt. Omdat SDW® niet altijd beschikbaar zal zijn, wordt het Adventus stuurbestand met de Stekkerdoos meegeleverd. (Op deze wijze wordt vermeden dat SDW® moet worden aangeschaft enkel en alleen voor het genereren van stuurbestanden).

Uitgangspunten en randvoorwaarden

De Stuurbestandgenerator is geprogrammeerd als een SDW® script. Dit houdt in dat voor het draaien van de Stuurbestandgenerator SDW® beschikbaar moet zijn evenals kennis over

het gebruik van sdw-scripts. Hiervoor wordt verwezen naar de handleiding van de SDW module SDWrite.

Het programma heet Crestbst.scr. Voor de installatie moet Crestbst.scr bekend worden gemaakt aan SDW®. Daarna kan Crestbst.scr vanuit SDW® worden gedraaid. (Overigens geldt als randvoorwaarde dat het Adventus gegevensmodel onder SDW® beschikbaar is.)

Het draaien van de Stuurbestandgenerator

Na installatie binnen de SDW® omgeving kan het programma worden gedraaid. Het programma vraagt de volgende informatie:

- naam en locatie van het stuurbestand
- versie Adventus gegevensmodel

Eindsituatie

Na het draaien van de Stuurbestandgenerator is het Adventus-stuurbestand aanwezig, dat als invoer kan dienen voor de generator van het uitwisselingsformaat.

Opmerkingen

1. De Stuurbestandgenerator is batch-georiënteerd. Dit houdt in dat (afgezien van enkele vragen bij de start) geen vragen meer gesteld worden aan de gebruikers en het functioneren dus niet meer kan worden beïnvloed.
2. De Stuurbestandgenerator wordt geprogrammeerd met behulp van SDW scripts en kan alleen worden gedraaid als SDWrite is geïnstalleerd.
3. De naam van het SDW-model en de datum waarop het stuurbestand is gegenereerd wordt opgenomen in het stuurbestand. Er wordt een apart veld in het stuurbestand vrijgehouden voor het handmatig toevoegen van het Adventus-versienummer.
4. Het versienummer van het bij het inpakken gebruikte classificatiestelsel, zoals bepaald door de beheersorganisatie ervan, moet worden opgegeven.

3.3 Adventus stuurbestand

In dit stuurbestand worden eerst alle voorkomende attributen beschreven en daarna alle entiteittypen. In het stuurbestand zijn ook enige administratieve gegevens opgenomen zoals een datum waarop het stuurbestand is gegenereerd, de gebruikte versie van Adventus en dergelijke. Een detailbeschrijving (het formaat) wordt gegeven in hoofdstuk 8

Opmerking

Het Adventus stuurbestand kan met een willekeurige teksteditor, waarmee ASCII-bestanden kunnen worden aangemaakt, worden gewijzigd.

3.4 Bilateraal stuurbestand

Het bilateraal stuurbestand kent exact hetzelfde formaat als het Adventus stuurbestand. In dit bestand kunnen aangepaste attributen en entiteittypen, die bekend zijn in Adventus worden opgegeven. Teven kunnen applicatie gebonden attributen en entiteittypen worden opgenomen. Ook de waardenreeksen van de Stekkerdoos worden in dit bestand opgegeven. Een detailbeschrijving van waardenreeksen wordt gegeven in hoofdstuk 2.

Als een entiteittype van Adventus in het bilaterale bestand wordt geherdefinieerd, dan geldt deze nieuwe definitie in plaats van de Adventus definitie.
Als een attribuut in het bilaterale stuurbestand wordt geherdefinieerd, dan geldt de nieuwe definitie voor alle entiteittypen (uit Adventus en bilateraal) waarin het betreffende attribuut wordt gebruikt.

Opmerkingen

1. Bij tegenstrijdigheden tussen de beide stuurbestanden krijgen de bilaterale afspraken de prioriteit boven de definities van Adventus.
2. Het bilaterale stuurbestand kan met een willekeurige teksteditor waarmee ASCII-bestanden kunnen worden aangemaakt, worden aangemaakt en/of gewijzigd.
3. De waardereeksen kunnen in ook in het stuurbestand worden gedefinieerd. Zie eveneens hoofdstuk.. voor een beschrijving en uitleg van waardenreeksen in de Stekkerdoos. Bekijk
4. ook het meegeleverde bilaterale stuurbestand als een voorbeeld.

3.5 Generator voor het uitwisselingsformaat

De generator is een zelfstandig programma met de naam CREUWF (creëer uitwisselingsformaat)

Wanneer gebruiken

Het is aan te bevelen om per applicatie waarin de Stekkerdoosfuncties worden gebruikt een bestand met het uitwisselingsformaat te maken. Dat kan door gebruik te maken van CREUWF en in ieder geval het Adventus stuurbestand als invoer te gebruiken. Afhankelijk van de applicatie kan al dan niet een bilateraal stuurbestand als tweede invoer worden gebruikt. Functies uit de Stekkerdoos gebruiken vervolgens het bestand met het uitwisselings-formaat.

Uitgangspunten en randvoorwaarden

CREUWF vereist tenminste als invoer het Adventus stuurbestand. De invoer van een bilateraal stuurbestand is optioneel.
De file-extensie van de uitvoer (= het bestand met het uitwisselings-formaat) wordt .uwf.
CREUWF is een zogenaamde *console application*. Bij het opstarten onder Microsoft Windows draait het in een DOS box.

Het draaien van CREUWF

Het programma is bekend als de file creuwf.exe. Start het programma onder Windows via dubbel-klik of via een short cut.
U wordt gevraagd de file namen op te geven:

- Adventus stuurbestand
- bilateraal stuurbestand, indien gewenst
- van het bestand voor het uitwisselingsformaat (de extensie .uwf wordt toegevoegd door het programma)

Bij fouten ontstaat de file error.log in de directory waar u CREUWF draait.

Het uitwisselingsformaat (=uitvoerbestand) copieert u vervolgens naar de gewenste directory voor gebruik door de Stekkerdoosfuncties die een uitwisselingsbestand gaan schrijven.

Logica
Het programma CREUWF vertaalt de stuurbestanden naar een uitwisselingsformaat. Onregelmatigheden die in de stuurbestanden worden geconstateerd worden gerapporteerd in de file error.log.

De logica is als volgt:
- lees de administratieve gegevens uit het Adventus stuurbestand
- lees de attributen uit het Adventus stuurbestand
- lees de entiteittypen uit het Adventus stuurbestand
- lees de administratieve gegevens van een eventueel aanwezig bilateraal stuurbestand
- lees de attributen uit een eventueel aanwezig bilateraal stuurbestand
- lees de entiteittypen vanuit het eventueel aanwezig bilateraal stuurbestand
- modificeer eventueel gegevenselementen zoals hieronder staat vermeld bij opmerkingen
- schrijf uitwisselingsformaat en gebruik daarbij de definities van attributen en entiteittypen uit het bilaterale stuurbestand als deze twee keer voorkomen

Opmerkingen
1. Bij gelijke UvW codes voor verschillende entiteiten of attributen wordt alleen de eerste opgenomen. De latere definities worden na een foutmelding genegeerd.
2. CREUWF is batch-georiënteerd. Dit houdt in dat (afgezien van enkele vragen bij de start) geen vragen meer gesteld worden aan de gebruikers en het functioneren dus niet meer kan worden beïnvloed.
3. Bij alfanumerieke velden in een stuurbestand worden eventueel opgegeven decimalen achter de punt genegeerd.
4. Datum velden zonder lengte worden verondersteld 8 posities lang te zijn.
5. Als in het stuurbestand een sleutelveld langer is dan 24 posities, dan wordt dit afgekapt op 24 posities.
6. Als geen type (alfanumeriek, numeriek of datum) is opgegeven in het stuurbestand dan wordt het veld verondersteld een alfanumeriek veld te zijn.
7. Als geen lengte is opgegeven in het stuurbestand dan wordt de lengte 8 verondersteld.
8. Entiteiten zonder sleutelattributen worden verondersteld waardereeksen te zijn. Hierbij moeten waarden zijn opgenomen voor de dimensies van de attributen.

3.6 Het uitwisselingsformaat

Het bestand met het uitwisselingsformaat bevat definities van alle attributen en entiteittypen uit de stuurbestanden in een zodanige vorm dat:
- alfanumerieke gegevens worden omgezet naar characterstrings van de lengte zoals opgenomen in het stuurbestand
- datumvelden worden opgenomen als characterstrings
- numerieke velden worden als volgt overgenomen:

- ◊ met decimale punt en totale lengte kleiner of gelijk 6 ⇒ Real*4
- ◊ met decimale punt en totale lengte kleiner of gelijk aan 15 ⇒ Real*8
- ◊ met decimale punt en totale lengte groter dan 15 ⇒ characterstring
- ◊ zonder decimale punt en een lengte kleiner of gelijk aan 9 ⇒ Integer*4
- ◊ zonder decimale punt en een lengte kleiner of gelijk aan 15 ⇒ Real*8
- ◊ zonder decimale punt en een lengte groter dan 15 ⇒ characterstring

In de te bouwen stekker moet rekening worden gehouden met deze omzettingen. Indien deze omzettingen naar integers en reals niet gewenst zijn dan moeten de gegevens in de stuurbestanden worden gedefinieerd als alfanumeriek. Deze omzetting is gedaan om het werken met vermenigvuldigingsfactoren en optelconstantes te vermijden. Bovendien kan met integers en reals worden gerekend in tegenstelling tot characterstrings. Echter, voor integer waarden van meer dan 9 posities is geen type beschikbaar in Fortran-77, evenals voor reals met meer dan 15 significante posities. Daarom worden deze gegevens beschouwd als characterstrings.

3.7 Stekkerdoosfuncties

3.7.1 Algemeen

Voor het PC platform is gekozen om de Stekkerdoosfuncties ter beschikking te stellen via een zogenoemde *DLL*. DLL staat voor **D**ynamic **L**ink **L**ibrary. De code van de bilblotheek wordt dynamisch aan het programma toegevoegd - dus pas tijdens het laden (bij executie) van het programma.

Bij het '*linken*' om een 'executable' te maken moet ook informatie van deze DLL worden toegevoegd. Door nu gebruik te maken van '*include files*' en een library (stkkrds.lib) is dit proces voor Microsoft of Digital Fortran transparant voor de gebruiker. Hij maakt gebruik van deze library om het verband te leggen tussen Stekkerdoosfuncties en de DLL. De library (stkkrds.lib) wordt meegeleverd, maar is specifiek voor Microsoft en Digital Fortran op een PC. Voor Visual Basic en C zijn aparte interfaces naar de DLL.

Specifieke informatie over de DLL wordt gegeven in hoofdstuk 4. DLL.

3.7.2 Performance

De Stekkerdoosfuncties kennen een reeks controles (met name via sleutels) om de integriteit van de entiteiten (records in de tabel) te waarborgen. Het houdt o.a. in dat sleutelwaarden worden gecontroleerd.
Verder kan willekeurig steeds een nieuw record aan de tabel worden toegevoegd. Tabellen hebben zo '*oneindige*' lengte.
Ook kunnen velden (attributen) van bestaande records worden overschreven.

Wanneer veel tabellen en/of veel records van tabellen naar de uitwisselingsfile worden geschreven kan de performance sterk teruglopen.

Om de performance op peil te houden zijn er mogelijkheden aan de Stekkerdoosfuncties toegevoegd. Allereerst kan de controle op sleutels worden uitgezet. Verder kan vooraf gekozen worden voor een vaste lengte van de tabel - de programmeur wordt dan volledig verantwoordelijk voor consistentie en integriteit van de tabellen en hun mogelijke lengte.

3.7.3 Functies

Onderstaand wordt een samenvatting van de Stekkerdoosfuncties gegeven. Hoofdstuk 6 geeft een gedetailleerde beschrijving.

De functies zijn te verdelen in de volgende categorieën:

- aanvragen (alloceren) van geheugen en het zetten van een aantal interne variabelen,
- openen en sluiten van een uitwisselingsbestand,
- creëren van tabellen,
- schrijven van velden uit een tabel naar een interne record buffer
- schrijven van de interne record buffer naar het uitwisselingsbestand (zowel update van bestaand record als insert van een nieuw record)
- lezen van een record uit een tabel in een interne record buffer
- lezen van velden van een tabel uit de interne record buffer
- schrijven van een kolom van een tabel naar het uitwisselingsbestand
- lezen van een kolom van een tabel van het uitwisselingsbestand
- creëren van een waardenreeks
- schrijven van attributen van een waardenreeks naar het uitwisselingsbestand
- lezen van attributen van een waardenreeks van het uitwisselingsbestand
- opvraag (inquire) functie over aantal tabellen en waardenreeksen op het uitwisselingsbestand en hun namen
- opvraagfunctie voor meer gegevens over de velden van tabellen
- opvraagfunctie voor meer gegevens over de attributen van waardenreeksen

Opmerkingen

1. De Stekkerdoos Water houdt een pointer bij naar het laatst gelezen of geschreven record van iedere tabel. Het is mogelijk het eerste of laatste record te lezen of op basis van sleutelwaarden. Bij het schrijven kan een insert achter laatste record worden gedaan of bestaand record worden overschreven.
2. De Stekkerdoos Water is ongevoelig voor verschillen tussen hoofdletters en kleine letters. Alles wat als character wordt opgegeven of gebruikt is in hoofdletters (wordt naar hoofdletters omgezet). Character- en sleutelvelden houden vanzelfsprekend wel hun opgegeven inhoud, dus met hoofd- en kleine letters
3. Alle door de Stekkerdoos Water gesignaleerde fouten en waarschuwingen worden naar een error file geschreven met een omschrijving. De filenaam is stekkerdoos.err en staat in de current directory van de applicatie (de stekker). Het ja/nee gebruiken van deze error file kan met een functie (*setval*) worden aangegeven.
4. Foutmeldingen afkomstig van Nefis (errorcode is negatief) behoren niet voor te komen. Indien dit wel gebeurt worden ze aan de gebruiker gemeld in de error file. Deze handleiding bevat alle Nefis foutmeldingen. Indien u er niet uitkomt, vraag dan hulp bij de helpdesk van Stekkerdoos Water.

5. Verwijderen van record is niet mogelijk, wel het veranderen van sleutelvelden en andere velden van bestaande records.
6. Te verwijderen velden kunnen als zodanig worden gemarkeerd met behulp van bijvoorbeeld een *was/wordt* status. Hiervoor is een apart attribuut (veld) met de naam WASWORDT van 8 karakters toegevoegd aan elke entiteit. Met deze status kan worden aangegeven of het betreffende record is gewijzigd, vervangen of vervallen. Dit veld kan door de stekkerbouwer op dezelfde manier worden gelezen en geschreven als de andere attributen. Gebruik bijvoorbeeld de volgende codering: "N" nieuw, "D" gedeeltelijke herziening, "H" gehele herziening en "V" vervallen.

3.8 Applicatie/stekker

Een applicatie kan gebouwd worden in bijvoorbeeld Fortran, C, Visual Basic of Delphi. Om die reden wordt een DLL beschikbaar gesteld. Details betreffende de DLL en het gebruik er van staan beschreven in hoofdstuk 4.

Voordat u start met het programmeren van een applicatie of stekker dienen de stappen of processen, zoals in de paragrafen hierboven beschreven, te zijn doorlopen. Met name een bestand met het uitwisselingsformaat is nodig bij het schrijven. Bij het lezen van het uitwisselingsbestand zou u kunnen volstaan met informatie die verkregen is via de opvraag (inquire) functies.
Anderzijds is het nodig dat de bouwer van applicaties/stekkers goede kennis heeft van de te gebruiken tabellen en waardenreeksen. Kennis van het Adventus- en het bilaterale stuurbestand is nodig, ook voor afmeting en type van velden in een tabel of de attributen van een waardenreeks. De stuurbestanden zijn met een ASCII-editor te bekijken.

Onderstaand worden de stappen beschreven, met gebruik van functienamen, om een uitwisselingsbestand te lezen of om er naar te schrijven. Er wordt niet ingegaan op alle details van het gebruik van deze functies. Zie hiervoor hoofdstuk 6.
Iedere applicatie moet beginnen met *allocatie* van geheugen. Dit geheugen is nodig voor de interne administratie en voor het gebruik van meerdere uitwisselingsbestanden. Hoeveel geheugen wordt gebruikt kan worden aangegeven met de functie SETVAL. Er zijn echter defaultwaarden opgenomen in de Stekkerdoos, waardoor het gebruik van SETVAL facultatief is.
==> Het eventule gebruik van SETVAL gaat vooraf aan het aanroepen van SWALLOC.

Het schrijven naar een uitwisselingsbestand bestaat uit:
- geheugenallocatie, functie SWALLOC
- openen uitwisselingsbestand met de functie OPNUWB
- creëren van een of meerdere tabellen met de functie CRETAB
- schrijven van waarden van velden naar een recordbuffer (die behoord bij een tabel) met de functie WRVLD. Ook sleutelwaarden worden met deze functie geschreven.
- schrijven van de record buffer naar het bestand met functie WRREC.
- afsluiten van het uitwisselingsbestand met functie CLSUWB.

Het lezen van een uitwisselingsbestand bestaat uit:
- geheugenallocatie, functie SWALLOC

- openen uitwisselingsbestand met de functie OPNUWB
- lezen van de record buffer van het bestand met functie RDREC.
- lezen van waarden van velden naar een recordbuffer (die behoord bij een tabel) met de functie RDVLD. Ook sleutelwaarden worden met deze functie gelezen.
- afsluiten van het uitwisselingsbestand met functie CLSUWB.

Ook kunnen kolommen van een tabel worden geschreven en gelezen. Hiervoor zijn de functies WRKOL en RDKOL. In plaats van de combinatie WRVLD en WRREC wordt dan WRKOL gebruikt. Voor het lezen geldt dat voor de overeenkomstige functies.

Per tabel is het aan te raden te kiezen voor het schrijven of lezen kolommen dan wel records met velden. Als via records is weggeschreven kan eventueel via kolommen worden teruggelezen en ook andersom.

Het schrijven van waardenreeksen gaat met de stappen:
- geheugenallocatie, functie SWALLOC
- openen uitwisselingsbestand met de functie OPNUWB
- creëren van een of meerdere waardenreeksen met de functie CREWRD
- schrijven van de waarden (=attributen van waardenreeks) naar het bestand met de functie WRWRD
- afsluiten van het uitwisselingsbestand met functie CLSUWB

Het lezen van waardenreeksen gaat met de stappen:
- geheugenallocatie, functie SWALLOC
- openen uitwisselingsbestand met de functie OPNUWB
- lezen van de waarden (=attributen van waardenreeks) naar het bestand met de functie RDWRD
- afsluiten van het uitwisselingsbestand met functie CLSUWB

Tabellen en waardenreeksen kunnen naar hetzelfde uitwisselingsbestand worden geschreven of er vanaf worden gelezen.

3.8.1 Algemene opmerkingen

1. Binnen een applicatie/stekker kan naar meerdere uitwisselingsbestanden worden geschreven of ervan worden gelezen.
2. In ieder bestand kunnen er meerdere verschillende tabellen aanwezig zijn. Van iedere tabel bestaat er slechts één met de naam zoals die in het stuurbestand is opgegeven.
3. Van eenzelfde waardenreeksdefinitie uit het stuurbestand kunnen meerdere voorkomens (=waardenreeksen) bestaan in een uitwisselingsbestand.
4. Er wordt één naam gebruikt voor het uitwisselingsbestand. Vanwege het gebruik van onderliggende Nefisfuncties ontstaan er twee bestanden met de extensies .def en .dat. Beide bestanden behoren bij elkaar.
5. Alle sleutelvelden zijn in Adventus van het type character (alfanumeriek). Dat geldt ook voor de Stekkerdoos Water en daarmee ook voor het bilaterale stuurbestand.

4 DLL

4.1 Algemeen

Levering van een DLL op het PC platform heeft enkele voordelen. Het belangrijkste is dat hiermee de Stekkerdoosfuncties ook beschikbaar komen voor C/C++, Visual Basic en Delphi.
Een ander voordeel is dat het onderhouden van slechts één DLL voor de PC forse kostenreducties geeft bij beheer en onderhoud. Levering van een DLL is voldoende omdat de huidige compilers van verschillende leveranciers voor het PC platform ook DLL's kennen. De DLL geeft een taal en compiler onafhankelijke toegang tot de Stekkerdoos-functies.

4.2 Digital Visual Fortran

Digital Visual Fortran 5.0 en hoger is de opvolger van Micrsoft Fortran PowerStation 4.0. De library file *"stkkrds.lib"* is een statische library en bevat specifieke informatie om relaties tussen de gebruikte routinenamen en de DLL-file vast te leggen. Deze library moet u opnemen in de lijst met libraries voor het *linken* van het programma.

Binnen Fortran kunt u direct de Stekkerdoos functies en hun parameters toepassen, zoals ze zijn beschreven in deze handleiding. Het gebruik van de library *stkkrds.lib* is voldoende. Bij het draaien van een applicatie-stekker moet de DLL beschikbaar zijn.

De functies voor lezen en schrijven van attributen en waardenreeksen moeten overweg kunnen met real double precision, integer en character type van een veld uit de tabel of attribuut van een waardenreeks. Bij de routines van de Stekkerdoos is hiermee rekening gehouden.
Indien gewenst kan de programmeur voor de Stekkerdoosfuncties zogenoemde *generic routines* maken met behulp van *interface* definitie en de specifieke routines voor de diverse datatypen.

4.3 Microsoft C/C++ en Visual Basic

4.3.1 Inleiding

Omdat de routines van deze DLL met Fortran zijn gemaakt, moet met een drietal zaken bij C en Visual Basic rekening worden gehouden:

- Alle genoemde parameters uit de beschrijving moeten altijd '*by reference*' worden aangeroepen, omdat Fortran dit ook doet.
- Een parameter van het Fortran type CHARACTER maakt dat er bij de aanroep op de stack een extra (voor Fortran verborgen) parameter bij komt. Hij is van het type '*by value*' en bevat de lengte van de character variabele.
- Het gebruik van C-strings vereist de nodige aandacht en wordt apart uitgelegd.

De twee eerste punten zijn verwerkt in de files *Stkkrds.h* voor C en *Func_def.bas* voor Visual Basic. Hierbij wordt aangetekend dat de integers in Fortran 4 bytes groot zijn.
De arrays en variabelen worden *by reference* doorgegeven. Voor C/C++ kan dat eenvoudig gebeuren.
Voor Visual Basic (VB) moet de eerste positie van de array worden doorgegeven en niet de array-naam. Dat komt omdat VB nog extra informatie voor de eerste positie heeft toegevoegd aan een array. Variabelen worden, zoals in VB gebruikelijk, al by reference doorgegeven.

Voor Visual Basic wordt de file naam van de DLL bij de functie-definitie gebruikt. Voor C/C++ is de bibliotheek *Stkkrds.lib* nodig om de relatie naar de DLL te leggen.
De juiste afbeelding van C-functie namen naar Fortran object namen in de DLL is geregeld in de file *Stkkrds.h*.

Details staan in het commentaar van de beide genoemde files.

4.3.2 Fortran CHARACTER

Een variabele in Fortran kan van het type CHARACTER zijn en heeft een lengte. Bijvoorbeeld:

```
CHARACTER*10      ::  naam
CHARACTER*256     ::  filenaam
```

De character variabelen worden niet afgesloten met een \0 zoals in C. De lengte bij Fortran geeft nl. aan hoeveel karakters we hebben.
In Fortran wordt bij het toekennen van een string aan een karakter-variabele deze aangevuld met spaties tot het einde. (dus `naam = 'pietje'` geeft dat karakters nr 7 t/m 10 van de variabele `naam` een spatie worden.)
Het is belangrijk hiermee rekening te houden in C en Visual Basic.

Een array van karakters is ook mogelijk en deze worden eveneens gebruikt bij de functies in de DLL van de Stekkerdoos Water. Een voorbeeld in Fortran is:

```
CHARACTER*12,  dimension(50)  ::  lijst
```

Dit geeft een array (met de naam `lijst`) van 50 plaatsen en ieder element van de array is 12 karakters groot. Alle karakters liggen achter elkaar, dus 600 in toaal.

Bij het gebruik van een array van karakters moet in C/C++ en in Visual Basic rekening worden gehouden met deze bovenstaande punten. Hieronder wordt dat verder uitgelegd.

4.3.3 Visual Basic en karakters

Visual Basic (VB) kent strings waarvan de karakters 16 bits gebruiken, de z.g.n. *Unicode*, in plaats van 8 bits karakters die Fortran kent. Er dient dus een omzetting plaats te vinden. Dat kan naar een array van het type byte. Bij de Stekkerdoos Water wordt er in de meegeleverde file *Func_def.bas* van uitgegaan dat de VB karakters in een variabele van het type byte zijn geplaats. Dat geld ook voor de karakter arrays.
Een simpele variabele is een array van bytes, bijvoorbeeld 10 of 256 lang. Evenzo moet een array van bytes voldoende lang zijn, bijvoorbeeld 600, om de Fortran karakter array te simuleren.
Bij de aanroep van een Stekkerdoos functie wordt dan de lengte van de string (=aantal bytes) meegegeven. Bij het doorgeven van een array van strings, bepaalt de karakterlengte

dan de indeling van de array-elementen die voor Fortran relevant is.
Op de diskette staat onder directory ...\Vb een voorbeeld.

4.3.4 C/C++ en karakters

De Fortran routines van de Stekkerdoos DLL kunnen uit de voeten met de C-strings als rekening wordt gehouden met ook het doorgeven van de string-lengte. Hiervoor kan een variable of constante worden gebruikt, maar beter is het om de functie *strlen* te gebruiken. Deze functie geeft de lengte van de string, zonder de \0 en dat is precies wat Fortran wil.
In de file *Stkkrds.h.* staan de prototypes van de DLL routines en daarbij is rekening gehouden met de extra parameter voor de karakter/string lengte.
In onderstaand stukje C-code is dat nog eens weergegeven.

```
#include <stdio.h>
#include <string.h>
#include <memory.h>

#include "Stkkrds.h"

      int error;
      char uwbnaam[20];
      char status[10];

      swalloc( &error ) ;

      (void)strcpy( uwbnaam, "uwbtest" ) ;
      (void)strcpy( status,  "OVERWRITE" ) ;

      opnuwb( &error, uwbnaam, strlen(uwbnaam), status,
              strlen(status) ) ;
```

Gebruik van een array van strings kan het beste door in C een `char` buffer van voldoende lengte te maken. Vul deze buffer eerst met spaties om problemen in de DLL routines (Fortran) te voorkomen. De functie *memset* kan hiervoor worden toegepast.
Plaats vervolgens op de goede posities de gewenste tekst, bijvoorbeeld met de functie *strncpy,* maar vermijdt dat de \0 mee wordt gecopieerd.
Bij aanroep van de DLL functie moet in dit geval ook een variabele of constante worden meegegeven voor de karakter/string lengte. Deze geeft de lengte in karakters van de array elementen aan.
Onderstaand een stukje code als voorbeeld.

```
#include <stdio.h>
#include <string.h>
#include <memory.h>

#include "Stkkrds.h"


#define  BUFLEN   5      /* aantal array elementen */
#define  BUFSIZE  12     /* lengte van een array element */
```

```
int error;
char uwbnam[20];
char status[10];
char uwfnam[64] ;
char tblnam[14] ;
int  dim ;
char vldnam[16] ;
int  aantal ;
char buffer[BUFLEN*BUFSIZE+1] ;

/* Alles vullen met spaties */
(void)memset( buffer+0, ' ' , BUFLEN*BUFSIZE ) ;

/* Laatste karakter \0,
   daarmee blijft het een C string */
buffer[BUFLEN*BUFSIZE] = '\0'  ;

/* Vul de buffer, gebruik maximaal 12 (=BUFSIZE)
karakters, minder mag ook */
strncpy( buffer+0,  "ABCDEFGHIJKL" , 12 ) ;
strncpy( buffer+12, "1234567890ab" , 12 ) ;
strncpy( buffer+24, "ABCDE"        , 5  ) ;
strncpy( buffer+36, "0123"         , 4  ) ;
strncpy( buffer+48, "OPQRSTUVWXYZ" , 12 ) ;

(void)strcpy( uwfnam, "sobek.uwf" ) ;
(void)strcpy( tblnam, "XDW_GC_BILA" ) ;
dim = 100 ;

cretab( &error, uwbnam, strlen(uwbnam), tblnam,
     strlen(tblnam), uwfnam, strlen(uwfnam),
     &dim ) ;


(void)strcpy( vldnam, "xdwdefid" ) ;
aantal = BUFSIZE ;

/* De karakter buffer bevat in de juiste volgorde de
karakters (strings) en
wordt aangeroepen met de juiste afmeting van de buffer
elementen */

wrkol_sc( &error, uwbnam, strlen(uwbnam), tblnam,
          strlen(tblnam), vldnam, strlen(vldnam),
          &aantal, buffer, BUFSIZE )  ;
```

Op de diskette onder directory ...\C_code staat een (ander) voorbeeld programma.

5 Installatie

5.1 Systeemeisen

Voor een goede werking van de Stekkerdoos Water gelden de volgende systeemeisen:
Voor het genereren van een GW'96 stuurbestand is nodig:

- een geïnstalleerde versie van GW'96,
- SDW-Workstation met de module DataModelling (DM) versie 3.1* of 3.2* (SDW kan worden geleverd door CAP-Gemini),
- de verdere systeemeisen (hardware en dergelijke) zijn afhankelijk van de versie van SDW.

Deze informatie kan worden opgevraagd bij CAP-Gemini.

Voor het genereren van het bestand met het uitwisselingsformaat (de *.uwf file) is nodig:

- het GW96-stuurbestand,
- indien gewenst een bilateraal stuurbestand,
- de generator, het programma CREUWB.

Voor de bouw van een applicatie/stekker gelden de volgende eisen:

- een bestand met het uitwisselingsformaat,
- een 32-bits Windows computer (Windows NT of Windows 95) met voldoende geheugen (bij voorkeur 64 Mbyte of meer)
- een 32-bits versie van een compiler of programmeeromgeving die overweg kan met DLL's.

5.2 Platforms en portabiliteit

De Stekkerdoos Water is zoveel mogelijk platform-onafhankelijk ontwikkeld. De ontwikkeling heeft plaatsgevonden binnen de 32-bits Windows-omgeving met gebruik van de Digital Fortran 5.0 (opvolger van Microsoft Fortran Power Station 4.0). Dit is een Fortran 90 compiler.
Bij gebruik op andere platforms moet zowel de broncode van NEFIS als van de Stekkerdoos Water naar het gewenste platform (met compiler) worden *'geporteerd'* en worden *'gecompileerd'*.

5.3 Installatie-instructie

De bestanden worden geleverd als een *'self extracting'* file op diskette. De filenaam is stkkrds_v3.exe

Er dient een directory bijvoorbeeld *\Stekkerdoos* te worden aangemaakt op de schijf waar de Stekkerdoos gebruikt gaat worden.

De installatie gebeurt dan door het uitvoeren van het commando *stkkrds_v3.exe* in de genoemde directory.

U dient erop te letten dat een evenutele directory stkkrds van versie 2.0 van de Stekkerdoos Water wordt verwijderd.

5.4 Programmeur instructies

Er wordt vanuit gegaan dat programmeurs die de Stekkerdoos Water gaan gebruiken voldoende op de hoogte zijn van de Fortran- of C-compiler of Visual Basic binnen hun organisatie. De programmeur kan de diverse files, zoals hierboven genoemd bij de DLL uitleg in hoofdstuk 4, toepassen bij het bouwen van een applicatie-stekker. De beperking is dat deze specifiek voor de Microsoft en/of Digital compilers bestemd zijn

In hoofdstuk 4 is uitvoerig ingegaan op het gebruik van de DLL, voor Fortran en C, maar ook voor Visual Basic 5.0.

5.5 Overzicht files op diskette

De distributiediskette bevat de volgende directories en bestanden:

5.5.1 Bibliotheek bestanden

De volgende bibliotheek bestanden staan in de directory ...\lib:

Stkkrds.dll	De DLL van de Stekkerdoos Water
Stkkrds.lib:	Een aanvullende library voor gebruik van de DLL bij de Microsoft of Digital compilers voor Fortran of C.
Dforrt.dll	Een extra DLL die ook gebruikt wordt en geïnstalleerd dient te worden.

5.5.2 Documentatie

Separaat zijn de gebruikershandleiding en het functioneel ontwerp beschikbaar. Indien er aanvullende informatie/documentatie beschikbaar komt, wordt dat in de directory ...\doc opgenomen. Hierbij komt de file Readme.txt waarin aanwijzingen en uitleg staan.
Op dit moment is de electronische versie van de handleiding daar geplaatst.

5.5.3 C-interface

Voor het schrijven van C-programma's is een C-header file beschikbaar. De directory bevat ook een voorbeeld in C en aanvullende files voor het voorbeeld.
In hoofdstuk 4 is meer uitgelegd over gebruik an de DLL en C/C++ en de volgende files staan in de directory ...\C_code:

Stkkrds.h	Header file met de prototypes van de DLL functies, gebaseerd op Micrsoft C/C++

Voorb_1.c	Een C voorbeeld programma
Test.uwf	De z.g.n. *formaat* file nodig bij het draaien van de test

We willen opmerken dat voor andere C compilers mogelijk andere interfaces en aanvullende code nodig zijn. Hiervoor zullen nieuwe directorynamen worden toegevoegd tijdens beheer en onderhoud. De directories zullen dan een file readme.txt bevatten met informatie over de betreffende directory.

5.5.4 Visual Basic

Er is een directory ...\Vb en hierin staan:

Func_def.bas	De interface definities van alle Stekkerdoosfucnties uit de DLL naar Visual Basic versie 5.0 van Microsoft.
Stekker.bas	Een voorbeeld in Visual Basic
Test.uwf	De z.g.n. *formaat* file nodig bij het draaien van de test

In hoofdstuk 4 wordt meer uitgelegd over het gebruik van de DLL en Visual Basic.

5.5.5 Voorbeelden

De voorbeelden uit de handleiding zijn beschikbaar als vijf files in de directory ...\Voorb.
In de directory staat een file Readme.txt met meer aanwijzingen. De file Test.uwf is nodig bij het draaien van de tests. Er staan ook output files om de eigen resultaten, na compileren en linken, te kunnen vergelijken.

De voorbeelden zijn gebaseerd op Digital Fortran versie 5.0.

5.5.6 Executables en SDW-scripts

De volgende executables en scripts staan op directory ...\Bin

Crestbst.scr:	Een SDW script voor het genereren van een stuurbestand vanuit een SDW versie van GW'96. (ASCII bestand)
Creuwb.exe:	Het programma CREUWB voor het omzetten van een stuurbestand naar een bestand met het uitwisselingsformaat.

5.5.7 Documentatie

Separaat is deze gebruikershandleiding beschikbaar. Daarnaast zijn ook het Functioneel Ontwerp en het Technisch onwerp in een rapport beschikbaar voor meer achtergrond informatie.

5.5.8 Stuurbestand Adventus (GW96) en voorbeeld bilateraal bestand

Het stuurbestand van Adventus (GW96) staat in de directory \stkkrds\stuur. In deze directory is ook een bilateraal stuurbestand opgenomen (filenaam is bilateraal).

GW96SBST: Een door CRESTBST gegenereerd stuurbestand met enkele noodzakelijke handmatige aanpassingen. (ASCII bestand) (Deze aanpassingen zijn er voor om te vermijden dat CREUWF halverwege de executie stopt. Zie voor details de technische documentatie).

BILASBST: Een voorbeeld van een bilateraal stuurbestand met tabellen en waardenreeksen. Dit stuurbestand wordt via het uitwisselingsformaat ook gebruikt bij de voorbeelden.

De files staan in de directory ...\Stuur op de diskette

6 Beschrijving van de functies

6.1 Overzicht en algemene opmerkingen

Alle beschikbare functies staan in onderstaand overzicht. De details en het gebruik worden in paragraaf 6.3 per functie beschreven.

	Functionaliteit
swalloc	alloceren van geheugen
setval	set-function om een aantal waarden/parameters te kunnen instellen
opnuwb	openen uitwisselingsbestand
clsuwb	sluiten uitwisselingsbestand
cretab	maken van een tabel
wrvld	schrijven van een veld
wrrec	schrijven van de recordbuffer
rdrec	vullen (lezen) van de recordbuffer
rdvld	lezen van een veld
wrkol	schrijven van een kolom
rdkol	vullen (lezen) van een kolom
crewrd	maken van een waardenreeks
wrwrd	schrijven van een waardenreeks(attribuut)
rdwrd	vullen (lezen) van waardenreeks(attribuut)
inquwb	opvraagfunctie welke tabellen en waardenreeksen er op een bestand staan
inqtbl	opvraagfunctie over specifieke informatie van één tabel en zijn velden
inqwrd	opvraagfunctie over specifieke informatie over één waardenreeks en zijn attributen

De namen van de bestanden, van tabellen, velden en waardenreeksen kunnen zowel in kleine letters als hoofdletters worden gegeven bij de aanroep van functies. Deze namen wordt altijd omgezet naar hoofdletters.

Bij de beschrijving van de parameters bij de functies worden opgegeven:

- de naam van de parameter,
- zijn type,
- de aanduiding In of Out voor input respectievelijk output parameter.

De type aanduiding kan aangeven Character*(*). Hiermee wordt bedoeld dat er een character variabele (Fortran), character constante of string wordt meegegeven. De lengte ervan (het aantal karakters) is vrij.
Bij character*(*) als input wordt de lengte van de aangeboden character variabele of string gebruikt tot ten hoogste het aantal karakters dat intern wordt gebruikt. Bijvoorbeeld voor een veldwaarde of sleutelwaarde of naam van tabel of waardenreeks.
Bij character*(*) als output parameter moet de programmeur ervoor zorgen dat de aangeboden ruimte voor character variabele of string tenminste lang genoeg is om de output

te ontvangen. Is dit niet het geval dan zal dat zeker tot (grote) problemen in de applicatie leiden omdat er dan geheugen wordt overschreven.

In hoofdstuk 4 over de DLL is reeds aangeven hoe vanuit andere talen dan Microsoft of Digital Fortran met character parameters voor deze DLL functies moet worden omgegaan.

6.2 Foutmeldingen

Het resultaat van alle subroutines wordt teruggegeven middels de parameter *error*. Het is zeer aan te bevelen om op de waarde van *error* te testen teneinde een goede werking van de applicatie/stekker te waarborgen.

De waarden van *error* kunnen zijn:

error is gelijk aan 0	De functie is correct uitgevoerd.
error groter dan 0	Er is een fout tijdens uitvoering. Het foutnummer correspondeert met het nummer zoals dat vermeld staat in de error file.
error kleiner dan 0	Er is een fout opgetreden tijdens lezen/schrijven van het uitwisselings-bestand. Het is een NEFIS-fout.

Het foutmeldingenbestand heeft een vaste naam, te weten stekkerdoos.err en dit bestand komt in de current directory van de applicatie/stekker te staan.
Het bestand bevat de foutmeldingen die afkomstig zijn van de Stekkerdoosfuncties, met per foutmelding:

1. de naam van functie waarin de fout optreedt,
2. het foutnummer, in deze gevallen positief,
3. een omschrijving.

De foutmeldingen, welke afkomstig van zijn van de onderliggende NEFIS-functies worden ook naar het foutmeldingenbestand geschreven met:

1. de naam van functie waarin de fout optreedt,
2. het foutnummer, in deze gevallen negatief.

De omschrijving van de NEFIS-fouten zijn opgenomen in de gebruikershandleiding. Een NEFIS-fout behoort niet voor te komen. Is dit toch het geval, dan kunt u mogelijk met de foutmelding zelf traceren waarom de NEFIS-fout optreedt bij het lezen van of schrijven naar het uitwisselingsbestand. Komt u er niet uit dan kunt u de Stekkerdoos helpdesk raadplegen.

Per applicatie/stekker wordt één foutmeldingenbestand gemaakt. Een reeds bestaand foutmeldingenbestand wordt overschreven.
Bij foutmeldingen dient u te bedenken dat er in de applicatie meerdere uitwisselingsbestanden geopend kunnen zijn. Een fout zal meestal bij een bepaald bestand optreden.

Het al dan niet schrijven van de foutmeldingen naar een bestand kan worden opgegeven met de functie SETVAL. De default situatie is, dat er wèl naar het foutmeldingenbestand wordt geschreven.

6.3 Detail beschrijving van de functies

6.3.1 SWALLOC

Aanroep: SWALLOC (errror)

error	**Integer**	**Out**
	zie paragraaf 6.2.	

Met deze functie wordt geheugen voor de Stekkerdoosfuncties aangevraagd. Deze functie **moet** éénmalig, (meestal) als eerste Stekkerdoosfunctie van een applicatie of stekker worden aangeroepen. Soms wordt hij vooraf gegaan door een of meerdere aanroepen van SETVAL, zie hieronder.

De omvang van het geheugen wordt met name bepaald door het aantal gelijktijdig openstaande uitwisselingsbestanden en het maximaal aantal tabellen + waardenreeksen op de uitwisselingsbestanden. Deze waarden staan default op respectievelijk 5 en op 60.
Met de functie SETVAL kunt u als gebruiker deze waarden opgeven/aanpassen. Het spreekt vanzelf dat u deze functie dan eerst moet aanroepen en daarna de functie SWALLOC.

6.3.2 SETVAL

Aanroep: SETVAL (error, varnaam, value)

Met deze functie kan de gebruiker een aantal variabelen zetten. Voor iedere te veranderen waarde is er de aanroep van deze functie met een karakerstring en de bijbehorende waarde. Voor het aantal tabellen en waardenreeksen per bestand en het aantal uitwisselingsbestanden worden maxima opgegeven.

error **Integer** **Out**
zie paragraaf 6.2.

varnaam **Character*(*)** **In**
karakterstring om aan tegeven welke variabele een waarde krijgt.
mogelijke waarden zijn:

AANTAL_TABELLEN	(max. tabellen + waardenreeksen samen)
AANTAL_UWB	(max. uitwisselingsbestanden)
FOUT_REGISTRATIE	(ja of nee meldingen naar file error.log)

value **Integer** **In**
waarde in combinatie met de *varnaam*, die aangeeft:
maximum aantal tabellen + waardenreeksen per bestand
maximum aantal openstaande uitwisselingsbestanden
value=1 wel meldingen schrijven; value=0 geen meldingen schrijven

Binnen de Stekkerdoos Water staat de waarde voor AANTAL - UWB default op 5 en voor AANTAL-TABELLEN default op 60.

6.3.3 OPNUWB

Aanroep: OPNUWB (error, uwbnaam, status)

Met deze functie wordt een uitwisselingsbestand geopend. Dit gebeurt op basis van:

1. de naam van het uitwisselingsbestand (uwbnam)
2. de status (status)

error **Integer** **Out**
zie paragraaf 6.2.

uwbnam **Character*(*)** **In**
de naam van het uitwisselingsbestand is ZONDER extensie, maximaal 256 karakters
Het is mogelijk om gelijktijdig meerdere uitwisselingsbestanden te gebruiken. Dat wil zeggen, meerdere malen aanroep van deze open-functie en later de *close-functie*. Bij alle functies van de Stekkerdoos Water wordt vervolgens de gewenste naam voor het uitwisselingsbestand opgegeven.

status **Character*(*)** **In**
met status wordt bepaald hoe en bestand wordt geopend.
status = 'NEW'
betekent dat een nieuw uitwisselingsbestand wordt geopend. Het bestand mag nog NIET bestaan. Van het uitwisselingsbestand kan zowel worden gelezen als geschreven.
status = 'OVERWRITE'
betekent dat een nieuw uitwisselingsbestand wordt geopend waarbij een eventueel bestaand uitwisselingsbestand zonder melding vooraf wordt overschreven. Van het uitwisselingsbestand kan zowel worden gelezen als geschreven.
status = 'READ'
betekent dat een bestaand uitwisselingsbestand wordt geopend om te lezen.
status = 'UPDATE'
betekent dat een bestaand uitwisselingsbestand wordt geopend om te lezen of te schrijven.

6.3.4 CLSUWB

Aanroep: CLSUWB (error, uwbnaam)

Met deze functie wordt een uitwisselingsbestand gesloten. Dit gebeurt op basis van:

1. de naam van het uitwisselingsbestand (uwbnam)

Tijdens het sluiten van een uitwisselingsbestand wordt de intene administratie op het bestand bijgewerkt en de NEFIS-buffers naar de file geschreven.
Het achterwege laten van deze functie heeft tot gevolg dat het uitwisselingsbestand corrupt zal zijn en niet meer te lezen valt.

error **Integer** **Out**
zie paragraaf 6.2.

uwbnam **character*(*)** **In**
de naam van het uitwisselingsbestand is ZONDER extensie, maximaal 256 karakters

6.3.5 CRETAB

Aanroep: CRETAB (error, uwbnam, tblnam, uwfnam, dim)

Met deze funtie wordt een tabel gedefinieerd op het uitwisselingsbestand op basis van de tabeldefinitie in het uitwisselingsformaat-bestand. De functie werkt op basis van:

1. de naam van het uitwisselingsbestand (uwbnam)
2. de naam van de te definieren tabel (tblnam)
3. de naam van het uitwisselingsformaat-bestand (uwfnam)
4. de lengte van de tabel (dim)

error **Integer** **Out**
zie paragraaf 6.2.

uwbnam **Character*(*)** **In**
de naam van het uitwisselingsbestand is ZONDER extensies, maximaal 256 karakters

tblnam **Character*(*)** **In**
de naam van de tabel en deze moet voorkomen op het uitwisselingsformaat-bestand

uwfnam **Character*(*)** **In**
de volledige naam van het uitwisselingsformaat-bestand (dus met eventuele extensie), maximaal 256 karakter

dim **Integer** **In**
hiermee wordt de lengte van de tabel gedefinieerd.
Indien vooraf niet bekend is wat de lengte (=aantal records) van de tabel zal worden, kan de waarde 0 worden gegeven.
In het geval de lengte vooraf niet bekend is, kan mogelijk toch een waarde hiervoor worden opgegeven. Dit verhoogt de performance. Echter, een éénmaal opgegeven lengte kan niet worden uitgebreid!

6.3.6 WRVLD

Aanroep: WRVLD (error, uwbnam, tblnam, vldnam ,vldwrd)

Met deze functie wordt een veld naar de recordbuffer geschreven. Deze functie is alleen voor de tabellen. Het schrijven gebeurt op basis van:

1. de naam van het uitwisselingsbestand (uwbnam)
2. de tabelnaam (op het uitwisselingsbestand) (tblnam)
3. de veldnaam (vldnam)
4. de veldwaarde vdlwrd)

error **Integer**
Zie paragraaf 6.2

uwbnam **Character*(*)**
De naam van het uitwisselingsbestand is ZONDER de toevoegingen .dat of .def

tblnam **Character*(*)**
De tabelnaam waarin het veld zich bevind.

vldnam **Character*(* In**
Het veld dat naar het recordbuffer geschreven moet worden.

vlwrd **?** **Out**
De veldwaarde zoals die naar het recordbuffer geschreven moet worden.

Omdat het type van *vldwrd* kan wisselen zijn er meerdere routines voor deze functie. Het zijn:

WRVLD_sc	voor sleutel en character velden
WRVLD_i	voor integer velden
WRVLD_r	voor real velden
WRVLD_d	voor double precision velden

Bij somminge compilers is het mogelijk om deze afzondelijke routines weer tot één generieke routine samen te voegen via een *interface definition*.

6.3.7 WRREC

Aanroep: WRREC (error, uwbnam, tblnam, wrtype, keychk, clrbuf)

Met deze functie wordt de interne recordbuffer naar een uitwisselingsbestand geschreven. Dit gebeurt op basis van:

1. de naam van het uitwisselingsbestand (uwbnam)
2. de tabelnaam (op het uitwisselingsbestand) (tblnam)
3. de wijze van schrijven (wrtype)
4. de vlag voor controle van sleutelwaarden (keychk)

error **Integer**
Zie paragraaf 6.2.

uwbnam **Character*(*)**
De naam van het uitwisselingsbestand is ZONDER de toevoegingen .dat of .def

tblnam **Character*(*)** **In**
de naam van de tabel waarin geschreven moet worden worden

wrtype **Character*(*)** **In**
met wrtype wordt aangegeven op welke wijze (en met welke snelheid) de gegevens naar het uitwisselingsbestand worden geschreven en het gebeurt in combinatie met parameter *keych.*

wrtype = 'INSERT'
betekent dat de gegevens uit het recordbuffer als nieuw record worden toegevoegd aan een tabel.
keychk = 'YES'
Betekent dat gecontroleerd zal worden of de sleutelwaarden uniek zijn. Als dat niet het geval wordt het record niet toegevoegd aan de tabel.
keychk = 'NO'
Betekent dat niet gecontroleerd zal worden of de sleutelwaarden uniek zijn en het record wordt zondermeer toegevoegd aan de tabel.

wrtype = 'UPDATE'
betekent dat de gegevens uit het recordbuffer het huidige record zullen overschrijven.
keychk = 'YES'
betekent dat gecontroleerd zal worden of de sleutelwaarden uniek zijn. Als dat niet het geval wordt het huidige record niet overschreven.
keychk = 'NO'

betekent dat niet gecontroleerd zal worden of de sleutelwaarden uniek zijn en wordt het record zondermeer overschreven met datgene wat in het recordbuffer staat.
LET OP: Alle velden die in de tabel (op het bestand) een waarde hadden zullen worden overschreven door de veldwaarden die nu in de record - buffer zitten. Dus informatei kan verloren gaan als niet eerst het record is gelezen van het bestand en dan een aantal velden zijn aangepast.

keychk **Character*(*)** **In**
Mogelijke waarden zijn 'YES' en 'NO'; zie verder *wrtype* hierboven.

clrbuf **Character*(*)** **In**
clrbuf='YES'
Na het schrijven worden de velden in de interne recordbuffer weer op de intiële waarden van blank of nul gezet.
Clbuf='NO'
Na het schrijven behouden de velden in de interne recordbuffer hun waarde
NB:
Alleen bij de eerste aanroep van WRREC per tabel en per uitwisselingsbestand staan er initiële waarden in de interne recordbuffer.

6.3.8 RDREC

Aanroep: RDREC (error, uwbnaam, tblnam, rdtype, slwrd)

Met deze functie wordt een record van een tabel gelezen en in de interne recordbuffer geplaatst. Dit gebeurt op basis van:

1. de naam van het uitwisselingsbestand (uwbnam)
2. de tabelnaam op het uitwisselingsbestand) (tblnam)
3. de wijze van lezen (rdtype)
4. de eventuele sleutelveldwaarden (slwrd)

error **Integer**
Zie paragraaf 6.2.

uwbnam **Character*(*)**
De naam van het uitwisselingsbestand is ZONDER de toevoegingen .dat of .def

tblnam **Character*(*)**
De tabelnaam waarvan een record moet worden gelezen.

rdtype **Character*(*)** **In**
deze bepaalt op welke wijze en met welke snelheid de gegevens van het uitwisselings-bestand worden gelezen

rdtype = 'SEARCH'
Betekent dat de sleutelveldwaarden (*slwrd*) gebruikt zullen worden om het record te vinden. Hiervoor zal vanaf het begin van de tabel worden gezocht.
slwrd moet voor ALLE sleutelvelden een zinvolle waarde hebben. Het is niet mogelijk om een beperkte set van sleutelvelden op te geven, omdat dit meerdere resultaten kan opleveren en er slechts één recordbuffer bestaat.
Indien het record niet wordt gevonden, komt er een foutnr (melding) en blijft de bestaande recordbuffer intakt.

rdtype = 'NEXT'
Betekent dat het volgende record in het recordbuffer wordt geplaatst.
Indien er geen 'volgend' record bestaat (bij het laatste record of een lege tabel) komt er een foutnr (melding) en wordt de functie verlaten. De bestaande recordbuffer blijft in dat geval intakt.

rdtype = 'FIRST'
Betekent dat het eerste record van de tabel wordt gelezen en in de recordbuffer geplaatst.
Indien er geen eerste record bestaat, (= lege tabel) komt er een foutnr (melding) en wordt de functie verlaten.

rdtype = 'LAST'
Betekent dat het laatste record van de tabel wordt gelezen en in de recordbuffer geplaatst.

Indien er geen laatste record bestaat, (= lege tabel) komt er een foutnr (melding) en wordt de functie verlaten.

slwrd **Character*(*)** **In**

Dit is een aaneengesloten characterstring, die gebruikt wordt bij het zoeken van een record in de tabel. Dus alleen zinvol bij rdtype = 'SEARCH'. De string bestaat uit alle sleutelvelden achter elkaar geplaatst, waarbij geldt:

- de volgorde waarmee de sleutelvelden in het uitwisselingsformaat staan, bepaalt de volgorde waarmee de velden in deze characterstring voorkomen
- bij het samenstellen van deze characterstring (character concatenatie) moet rekening worden gehouden met de lengte zoals die is opgegeven in het uitwisselingsformaat-bestand

6.3.9 RDVLD

Aanroep: RDVLD (error, uwbnam, tblnam, vldnam ,vldwrd)

Met deze functie wordt een veld uit de recordbuffer gelezen. De functie is alleen voor tabellen. Het lezen gebeurt op basis van:

1. de naam van het uitwisselingsbestand (uwbnam)
2. de tabelnaam (op het uitwisselingsbestand) (tblnam)
3. de veldnaam (vldnam)

Het resultaat van deze functie komt in parameter vldwrd

error **Integer**
Zie paragraaf 6.2.

uwbnam **Character*(*)**
De naam van het uitwisselingsbestand is ZONDER de toevoegingen .dat of .def

tblnam **Character*(*)**
De tabelnaam waarin het veld zich bevind.

vldnam **Character*(*)** **In**
Het veld dat uit de recordbuffer gelezen moet worden.

vldwrd **?** **Out**
De veldwaarde zoals die uit het recordbuffer wordt gelezen.

Omdat het type van *vldwrd* kan wisselen zijn er meerdere routines voor deze functie. Het zijn:

RDVLD_sc	voor sleutel en character velden
RDVLD_i	voor integer velden
RDVLD_r	voor real velden
RDVLD_d	voor double precision velden

Bij somminge compilers is het mogelijk om deze afzondelijke routines weer tot één generieke routine samen te voegen via een *interface definition.*

6.3.10 WRKOL

Aanroep WRKOL (error, uwbnam, tblnam, vldnam, nwrd, vldwrd)

Met deze functie wordt een kolom naar een tabel op een uitwisselingsbestand geschreven. Dit gebeurt op basis van:

1. de naam van het uitwisselingsbestand (uwbnam)
2. de tabelnaam (op het uitwisselingsbestand (tblnam)
3. de veldnaam (vldnam)
4. het aantal te schrijven veldwaarden (nwrd)
5. de veldwaarden (vldwrd)

Met deze functie wordt geen gebruik gemaakt van het record buffer; de veldwaarden worden rechtstreeks naar de tabel geschreven. Er vind geen controle plaats op het uniek zijn van sleutelveld-waarden, terwijl het schrijven per definitie begint op record 1 van de tabel.

error **Integer**
Zie paragraaf 6.2.

uwbnam **Character*(*)**
De naam van het uitwisselingsbestand is ZONDER de toevoegingen .dat of .def

tblnam **Character*(*)**
De tabelnaam waarin het veld zich bevind.

vldnam **Character*(*)** **In**
Het veld dat als kolom geschreven moet worden.

nwrd **Integer** **In**
Het aantal veldwaarden dat geschreven moet worden.

vldwrd **?** **In**
De array (=kolom) met veldwaarden zoals die naar de tabel geschreven wordt.

Omdat het type van *vldwrd* kan wisselen zijn er meerdere routines voor deze functie. Het zijn:

WRKOL_sc	voor sleutel en character velden
WRKOL_i	voor integer velden
WRKOL_r	voor real velden
WRKOL_d	voor double precision velden

Bij somminge compilers is het mogelijk om deze afzondelijke routines weer tot één generieke routine samen te voegen via een *interface definition*.

6.3.11 RDKOL

Aanroep: RDKOL (error, uwbnam, tblnam, vldnam, nwrd, vldwrd)

Met deze functie wordt een kolom van een tabel op een uitwisselingsbestand gelezen. Dit gebeurt op basis van:

1. de naam van het uitwisselingsbestand (uwbnam)
2. de tabelnaam (tblnam)
3. de veldnaam (vldnam)
4. het aantal te lezen veldwaarden (nwrd)
5.

Als resultaat worden teruggegeven de veldwaarden van een kolom en het aantal gelezen waarden.

Met deze functie wordt geen gebruik gemaakt van het record buffer; de veldwaarden worden rechtstreeks van de tabel gelezen en als resultaat doorgegeven aan de applicatie. Het lezen van een kolom start per definitie op record 1 van de tabel.

error **Integer**
Zie paragraaf 6.2.

uwbnam **Character*(*)**
De naam van het uitwisselingsbestand is ZONDER de toevoegingen .dat of .def

tblnam **Character*(*)**
De tabelnaam waarin het veld zich bevind.

vldnam **Character*(*)**
De veldnaam die (als kolom) gelezen moet worden.

nwrd **Integer** **In/Out**
In de input-fase bepaald *nwrd* de lengte van de array *vldwrd* waarin de veldwaarden opgeslagen moeten worden.
In de output-fase bevat *nwrd* het werkelijk gelezen aantal veldwaarden.
Als *nwrd* kleiner is dan het beschikbare aantal records van de tabel, wordt de kolom niet gelezen!

vldwrd **?** **Out**
Dit is de array (=kolom) met veldwaarden van een tabel, zoals die van het bestand is gelezen.

Omdat het type van *vldwrd* kan wisselen zijn er meerdere routines voor deze functie. Het zijn:

RDKOL_sc voor sleutel en character velden
RDKOL_i voor integer velden
RDKOL_r voor real velden
RDKOL_d voor double precision velden

Bij somminge compilers is het mogelijk om deze afzondelijke routines weer tot één generieke routine samen te voegen via een *interface definition*.

6.3.12 CREWRD

Aanroep: CREWRD (error, uwbnam, wrddef, wrdnam, uwfnam, dim)

Met deze funtie wordt een waardenreeks gedefinieerd op het uitwisselingsbestand op basis van de tabel-definitie in het uitwisselingsformaat-bestand. De functie werkt op basis van:

1. de naam van het uitwisselingsbestand (uwbnam)
2. de waardenreeks definitie (wrddef)
3. de naam van de te definieren waardenreeks (wrdnam)
4. de naam van het uitwisselingsformaat-bestand (uwfnam)
5. de lengte van de waardenreks (dim)

error **Integer** **Out**
Zie paragraaf 6.2.

uwbnam **Character*(*** **In)**
De naam van het uitwisselingsbestand is ZONDER de toevoegingen .dat of .def

wrddef **Character*(*)** **In**
De waardenreeks definitie, die moet voorkomen op het uitwisselingsformaat-bestand
Deze definitie wordt gebruikt voor de structuur van de waardenreeks.

wrdnam **Character*(*)** **In**
De naam van de waardenreeks, die uniek moet zijn
Onder deze naam is waardenreeks bekend op het uitwisselingsbestand.

uwfnam **Character*(*)** **In**
De volledige naam van het uitwisselingsformaat-bestand (dus met eventuele extensie), maximaal 256 karakter

dim **Integer** **In**
Deze geeft de lengte van de waardenreeks.
Het begrip lengte van een waardenreeks is beschreven in hoofdstuk..
De afmeting van de waardenreeks-attributen is vastgelegd in het uitwisselings-formaat-bestand.
Meestal zal de lengte van een waardenreeks bekend zijn. Indien deze vooraf niet bekend is, kan de waarde 0 worden gegeven. De eenmaal opgegeven lengte kan niet worden veranderd voor de gecreëerde waardenreeks.

6.3.13 WRWRD

Aanroep: WRWRD (error, uwbnam, wrdnam, attnam, nwrd, attwrd)

Met deze functie wordt een attribuut van een waardenreeks naar een uitwisselingsbestand geschreven. Dit gebeurt op basis van:

1. de naam van het uitwisselingsbestand (uwbnam)
2. de waardenreeksnaam (wrdnam)
3. de attribuutnaam (attnam)
4. het aantal te schrijven veldwaarden (nwrd)
5. de attribuutwaarden (attwrd)

error **Integer**
Zie paragraaf 6.2.

uwbnam **character*(*)**
De naam van het uitwisselingsbestand is ZONDER de toevoegingen .dat of .def

wrdnam **Character*(* In)**
De reeds eerder (CREWRD) gecreëerde waardenreeksnaam met daarin het attribuut met deze naam.

attnam **Character*(*) In**
Het attribuut dat naar uitwisselingsbestand geschreven moet worden.

nwrd **Integer In**
Dit is het aantal attribuutwaarden (ook wel lengte van de waardenreeks) dat geschreven moet worden. Variabel *nwrd* moet kleiner of gelijk zijn aan de lengte (*dim*) van de waardenreeks, zoals opgegeven bij CREWRD.
Het begrip lengte van een waardenreeks wordt uitgelegd in hoofdstuk ..

attwrd **? In**
Deze variabele (= array) bevat de waarden zoals die naar de tabel geschreven worden.
De array heeft aantal dimensies en afmetingen, die gelijk moeten zijn aan de dimensie en afmeting, zoals vastgelegd in het bilaterale stuurbestand. Variabele *nwrd* bepaald dan de laatste dimensie van deze array *attwrd*. Zie ook het voorbeeldprogramma met waardenreeksen.

Omdat het type van *attwrd* kan wisselen zijn er meerdere routines voor deze functie. Het zijn:
WRWRD_c voor character velden
WRWRD_i voor integer velden
WRWRD_r voor real velden
WRWRD_d voor double precision velden
Bij somminge compilers is het mogelijk om deze afzondelijke routines weer tot één generieke routine samen te voegen via een *interface definition*.

6.3.14 RDWRD

RDWRD (error, uwbnam, wrdnam, attnam, nwrd, attwrd)

Met deze functie wordt een attribuut van een waardenreeks op een uitwisselingsbestand gelezen. Dit gebeurt op basis van:

1. de naam van het uitwisselingsbestand (uwbnam)
2. de waardenreeksnaam (wrdnam)
3. de attribuutnaam (attnam)
4. het aantal te lezen attribuutwaarden (nwrd)

De gelezen attibuutwaarden staan in de array attwrd.

error **Integer**
Zie paragraaf 6.2.

uwbnam **Character*(*)**
De naam van het uitwisselingsbestand is ZONDER de toevoegingen .dat of .def

wrdnam **Character*(* In)**
De reeds eerder (CREWRD) gecreëerde waardenreeksnaam met daarin het attribuut met deze naam.

attnam **Character*(*) In**
Het attribuut dat van het uitwisselingsbestand gelezen moet worden.

nwrd **Integer** **In**
Dit is het aantal attribuutwaarden (ook wel lengte van de waardenreeks) dat gelezen moet worden. Variabel *nwrd* moet kleiner of gelijk zijn aan de lengte (*dim*) van de waardenreeks, zoals opgegeven bij CREWRD.
Het begrip lengte van een waardenreeks wordt uitgelegd in hoofdstuk ..

attwrd **?** **Out**
In deze variabele (= array) komen de waarden zoals die van het uitwisselingsbestand worden gelezen.
De array heeft het aantal dimensies en afmetingen, die gelijk moeten zijn aan de dimensie en afmeting, zoals vastgelegd in het bilaterale stuurbestand. Variabele *nwrd* bepaald dan de laatste dimensie van deze array *attwrd*. Zie ook het voorbeeldprogramma met waardenreeksen.
Het is de verantwoordelijkheid van de programmeur voldoende ruimte beschikbaar te stellen!

Omdat het type van *attwrd* kan wisselen zijn er meerdere routines voor deze functie. Het zijn:

RDWRD_c	voor character velden
RDWRD_i	voor integer velden
RDWRD_r	voor real velden
RDWRD_d	voor double precision velden

Bij somminge compilers is het mogelijk om deze afzondelijke routines weer tot één generieke routine samen te voegen via een *interface definition*.

6.3.15 INQUWB

Aanroep: INQUWB (error, uwbnaam, nr_tbl, lsttbl, nr_wrd, lstwrd)

Met deze functie wordt informatie opgevraagd over aantallen tabellen en waardenreeksen die op een uitwisselingsbestand staan . Dit gebeurt op basis van:

1. de naam van het uitwisselingsbestand (uwbnam)
2. dimensie array *lsttbl* (nr_tbl)
3. dimensie array *lstwrd* nr_wrd)

Het resultaat van deze functie is:

1. aantal tabellen (nr_tbl)
2. de lijst met tabelnamen (lsttbl)
3. aantal gecreëerde waardenreeks (nr_wrd)
4. de lijst met waardenreeks namen (lstwrd)

error **Integer**
Zie paragraaf 6.2.

uwbnam **Character*(*)**
De naam van het uitwisselingsbestand is ZONDER de toevoegingen .dat of .def

nr_tbl **Integer** **In/Out**
Bij input geeft nr_tbl de lengte van de array lsttbl waarin de tabelnamen teruggeven worden.
Bij output-bevat *nr_tbl* het werkelijk aantal tabellen op het uitwisselingsbestand. Als *nr_tbl* bij input kleiner is dan het aantal tabellen op het bestand, wordt de array *lsttbl* niet gevuld.

lsttbl **Character*(*)** **Out**
De lijst (array) met tabellennamen. De elementen van de array moeten voldoende karakterlengte (tenminste 13) hebben

nr_wrdInteger **In/Out**
Bij input geeft *nr_wrd* de lengte de array *lstwrd* waarin de namen van de gecreëerde waardenreeksen teruggegeven worden.
Bij output bevat *nr_wrd* het werkelijk aantal waardenreeksen op het uitwisselingsbestand. Als *nr_wrd* bij input kleiner is dan het aantal waardenreeksnamen op het bestand, wordt de array *lstwrd* niet gevuld.

lstwrd **Character*(*)** **Out**
De lijst (array) met waardenreeksnamen, zoals door de gebruiker gecreëerd. De elementen van de array moeten voldoende karakterlengte (tenminste 13) hebben

6.3.16 INQTBL

Aanroep: INQTBL (error, uwbnam, tblnam, nr_rec, bil, nr_vld, lst_vld, lsttyp, lstvnr)

Met deze functie wordt informatie opgevraagd over een specifieke tabel op een uitwisselingsbestand.
Dit gebeurt op basis van:

1. de naam van het uitwisselingsbestand (uwbnam)
2. de tabelnaam (tblnam)
3. dimensie van array's *lstvld, lsttyp* en *lstvnr.* (nr_vld)

Het resultaat van deze functie is:

1. het laatste record van de tabel
2. de bilateraal indicator (bil)
3. het aantal velden (nr_vld)
4. de lijst met veldnamen (lstvld)
5. de lijst met veldtypen (lsttyp)
6. de lijst met lengte van een kolom (plaats-nr van de laatste veldwaarde))

error **Integer**
Zie paragraaf 6.2.

uwbnam **Character*(*)**
De naam van het uitwisselingsbestand is ZONDER de toevoegingen .dat of .def

tblnam **Character*(*)** **In**
Tabel naam, kan bijvoorbeeld verkregen zijn met de functie INQUWB.

nr_rec **Integer** **Out**
Het aantal geschreven records van de tabel. Het aantal records wordt bepaald door het gebruik van functie WRREC, want dan zijn alle velden van een record beschreven (al dan niet met initiële of werkelijke waarden).
Wordt de functie WRKOL gebruikt dan staat niet éénduidig vast of alle kolommen zijn geschreven. Indien alle kolommen zijn weggeschreven, is het aantal records ook bekend en wel de lengte van de kolommen. De lengte van iedere kolom wordt apart gegeven in de array *lstvnr*.
Variabele nr_rec kan 0 zijn en er kunnen dan wel kolommen (velden) met een waarde voor de lengte van de kolom (plaats-nr van de laatste veldwaarde).

bil **Character*(*)** **Out**
Indicator of een tabel uit het bilaterale stuurbestand is gebruikt. Lengte is teminste 1 karakter.
bil = 'N' geen bilaterale tabel
bil = 'Y' wel bilaterale tabel

nr_vld **Integer** **In/Out**
Bij input geeft *nr_vld* de lengte van de array's *lstvld, lsttyp* en *lstvnr* waarin de informatie over velden wordt teruggegeven.
Bij output-is *nr_vld* het werkelijk aantal velden van de gevraagde tabel op het uitwisselingsbestand. Als bij input *nr_vldl* kleiner is dan aantal velden van de tabel, worden de array's *lstvld, lsttyp* en *lstvnr* niet gevuld.

lstvld **Character*(*)** **Out**
De lijst (array) met veldnamen. De elementen van deze array moeten voldoende karakterlengte (tenminste 8) hebben.

lsttyp **Character*(*)** **Out**
De lijst array met veldtypen. De elementen van deze array moeten tenminst 1 karakter groot zijn.
De volgende waarden kunnen worden teruggegeven:
'S' voor sleutel-veld
'C' voor character-veld
'I' voor integer-veld
'R' voor real-veld
'D' voor double precision-veld

lstvnr **Integer*(*)** **Out**
De lijst array met het plaatsnummer van de laatste veldwaarde (= de kolomlengte).
Zie ook de opmerkingen bij variabele *rc_nr*.

6.3.17 INQWRD

Aanroep: INQWRD (error, uwbnam, wrdnam, nr_wrd, bil, wrddef, attrdim, nr_attr, lstattr, lsttyp, lstanr)

Met deze functie wordt informatie opgevraagd over een specifieke tabel op een uitwisselingsbestand.
Dit gebeurt op basis van:

1. de naam van het uitwisselingsbestand (uwbnam)
2. de naam van de waardenreeks (wrdnam)
3. dimensie van array's *lstattr, lsttyp* en *lstanr.* (nr_attr)

Het resultaat van deze functie is:

1. de lengte van de waardenreeks (nr_wrd)
2. de bilateraal indicator (bil)
3. de naam van de waardenreeksdefinitie (wrddef)
4. de attribuutdimensies (attrdim)
5. het aantal attributen (nr_attr)
6. de lijst met veldnamen (lstattr)
7. de lijst met veldtypen (lsttyp)
8. de lijst met de lengte van een attribuut (lstanr)

error **Integer**
Zie paragraaf 6.2.

uwbnam **Character*(*)** **In**
De naam van het uitwisselingsbestand is ZONDER de toevoegingen .dat of .def

wrdnam **Character*(*)** **In**
De waardenreeksnaam, gecreëerd door de gebruiker en kan bijvoorbeeld verkregen zijn met de functie INQUWB.

nr_wrdInteger **Out**
De lengte van de waardenreeks. Het begrip lengte van waardenreeks is uitgelegd in hoofdstuk .. Is de waarde van *nr_wrd* gelijk aan nul, dan wil dat zeggen dat niet alle attributen van de waardenreeks naar het uitwisselingsbestand zijn gegeschreven.
De lengte van ieder afzonderlijk attribuut van de waardenreeks wordt gegeven in de array *lstanr*. Is deze nul, dan is het attribuut niet geschreven.

bil **Character*(*)** **Out**
Indicator of een tabel uit het bilaterale stuurbestand is gebruikt. Lengte is teminste 1 karakter.
bil = 'N' geen bilaterale tabel
bil = 'Y' wel bilaterale tabel

wrddef **Character** **Out**
Geeft de naam van de waardenreeksdefinitie uit het stuurbestand, waarop deze specifieke waardenreeks is gebaseerd (gecreëerd met functie CREWRD). De lengte van de karaktervariabele moet tenminste 13 zijn.

attrdim **Integer** **Out**
Een array waarin de dimensies en de afmeting van de attributen van deze specifieke waardenreeks staan. Het zijn de waarden die ook in het stuurbestand zijn gegeven. Een waarde van 0 wil zeggen dat deze dimensie niet wordt gebruikt. De array moet tenminste 4 lang zijn.

nr_attr **Integer** **In/Out**
Bij input geeft *nr_attr* de lengte van de array's *lstvld, lsttyp* en *lstanr,* waarin de informatie over attributen wordt terug gegeven.
Bij output-is *nr_vld* het werkelijk aantal attributen van de gevraagde waardenreeks op het uitwisselingsbestand. Als bij input *nr_vldl* kleiner is dan aantal attributen van de waardenreeks, worden de array's *lstvld, lsttyp* en *lstvnr* niet gevuld.

lstatttr **Character(*)** **Out**
De lijst (array) met attribuutnamen van de waardenreeks. De elementen van deze array moeten voldoende karakterlengte (tenminste 8) hebben.

lsttyp **Character*(*)** **Out**
De lijst (array) met veldtypen. De elementen van deze array moeten tenminst 1 karakter groot zijn.
De volgende waarden kunnen worden teruggegeven:
'C' voor character-veld
'I' voor integer-veld
'R' voor real-veld
'D' voor double precision-veld

lstanr **Integer*(*)** **Out**
De (lijst) array met de lengte van het attribuut van de waardenreeks. Lengte van waardenreeks is uitgegelegd in hoofdstuk .. Is de waarde 0, dan is dit attribuut niet geschreven. Zie ook de opmerkingen bij variabele *nr_wrd.*

7 Voorbeelden

Dit hoofdstuk geeft een aantal voorbeelden, en wel:

- schrijven van tabel,
- lezen van tabel,
- schrijven van waardenreeks,
- opvraag (inquire) functies.

De voorbeelden zijn in Fortran en daarbij zijn de specifieke functienamen gebruikt in verband met het lezen van en schrijven naar het uitwisselingsbestand van velden en waardenreeksattributen van het type *character, real, integer* en *double precision.*

De voorbeelden zijn ook beschikbaar op de installatiediskette

7.1 Schrijven van een tabel

```
      program voorb_1

c Voorbeeld programma, waarbij een tabel wordt geschreven naar
c het uitwisselingsbestand.
c De tabel bevat alle typen velden en is gedefineerd op een
c bilateraal stuurbestand.
c
      integer         error, dim, i
      character*20    uwbnam
      character*20    status
      character*20    tblnam
      character*20    uwfnam

c Bij deze declaratie van variabelen is rekening gehouden met
c de veld (attribuut) definitie op het bilaterale bestand.
c
      character*8     sv1
      character*24    sv2
      character*24    cv1
      character*1     cv2
      character*256   cv3
      integer         iv
      real            rv
      double precision dv

c Openen van file voor uitvoer van meldingen

      open (unit=9, file='voorb1.txt')

c Geef aantal initiele waarden, waaronder de filenamen
      uwbnam = 'voorb1'
      status = 'OVERWRITE'
      uwfnam = 'test.uwf'
      error  = 0

c Onderstaand de stappen, zoals ze ook in de handleiding zijn
c beschreven.

      write (9,'(a)') 'Start Voorbeeld 1'
```

```
      call SWALLOC (error)
      if (error .ne. 0) goto 9000

      call OPNUWB( error, uwbnam, status)
      if (error .ne. 0) goto 9000

c We werken hier met een vaste (geschatte) lengte van de tabel.
c De lengte hoeft niet volledig te worden gebruikt.
c De variabele dim (voor de lengte) zou ook 0 kunnnen zijn.
c De tabelnaam komt van het bilaterale bestand.
c
      dim    = 1000
      tblnam = 'ENSCIRD'

      call CRETAB( error, uwbnam, tblnam, uwfnam, dim)
      if (error .ne. 0) goto 9000

c Het karakterveld "cv2" krijgt een vaste waarde.
c Het karakterveld "cv3" krijgt een initiele waarden en
c wordt vervolgens binnen de loop gevuld.
c De twee sleutelvelden worden binnen de loop gevuld, waarbij
c op alle posities een karakter wordt gezet.
c Het vullen gebeurt met write statement.
c
c Dan wordn binnen de loop alle velden van de tabel naar het
c bestand geschreven, totaal 600 records.
c
      cv2 = '9'
      cv3(1:) = ' '

      do i=1,600

         write (sv1,'(i8.8)') i
         write (sv2,'(i24.24)') i
         write (cv1,'(i24.24)') i
         write (cv3(1:10),'(i10.10)') i
         iv = i
         rv = REAL(i)
         dv = DBLE(i)

c        Vul alle velden
c
         call WRVLD_sc( error, uwbnam, 'ENSCIRD', 'sv1' ,sv1)
         if (error .ne. 0) goto 9000
         call WRVLD_sc( error, uwbnam, 'ENSCIRD', 'sv2' ,sv2)
         if (error .ne. 0) goto 9000
         call WRVLD_sc( error, uwbnam, 'ENSCIRD', 'cv1' ,cv1)
         if (error .ne. 0) goto 9000
         call WRVLD_sc( error, uwbnam, 'ENSCIRD', 'cv2' ,cv2)
         if (error .ne. 0) goto 9000
         call WRVLD_sc( error, uwbnam, 'ENSCIRD', 'cv3' ,cv3)
         if (error .ne. 0) goto 9000
         call WRVLD_i ( error, uwbnam, 'ENSCIRD', 'iv1' ,iv)
         if (error .ne. 0) goto 9000
         call WRVLD_i ( error, uwbnam, 'ENSCIRD', 'iv2' ,iv)
         if (error .ne. 0) goto 9000
         call WRVLD_i ( error, uwbnam, 'ENSCIRD', 'iv3' ,iv)
         if (error .ne. 0) goto 9000
         call WRVLD_i ( error, uwbnam, 'ENSCIRD', 'iv4' ,iv)
         if (error .ne. 0) goto 9000
         call WRVLD_r ( error, uwbnam, 'ENSCIRD', 'rv1' ,rv)
         if (error .ne. 0) goto 9000
         call WRVLD_r ( error, uwbnam, 'ENSCIRD', 'rv2' ,rv)
```

```
            if (error .ne. 0) goto 9000
            call WRVLD_r ( error, uwbnam, 'ENSCIRD', 'rv3' ,rv)
            if (error .ne. 0) goto 9000
            call WRVLD_r ( error, uwbnam, 'ENSCIRD', 'rv4' ,rv)
            if (error .ne. 0) goto 9000
            call WRVLD_r ( error, uwbnam, 'ENSCIRD', 'rv5' ,rv)
            if (error .ne. 0) goto 9000
            call WRVLD_d ( error, uwbnam, 'ENSCIRD', 'dv1' ,dv)
            if (error .ne. 0) goto 9000
            call WRVLD_d ( error, uwbnam, 'ENSCIRD', 'dv2' ,dv)
            if (error .ne. 0) goto 9000
            call WRVLD_d ( error, uwbnam, 'ENSCIRD', 'dv3' ,dv)
            if (error .ne. 0) goto 9000
c
c           Schrijf nu het record
c
            call WRREC( error, uwbnam, 'ENSCIRD', 'INSERT', 'NO', 'NO')
            if (error .ne. 0) goto 9000

      enddo

 9000 continue
c
c Sluit het uitwisselingsbestand, ook in geval van een fout.
c Schrijf eerst het foutnr (of errror = 0 van de laatste correcte
c actie).
c
      write (9,'(a,i6)') 'Error = ', error
      call CLSUWB( error, uwbnam)
      write (9,'(a,i6)') 'Einde Voorbeeld 1: error = ', error

      close (9)

      stop

      end
```

7.2 Lezen van een tabel

```
      program voorb_2

c Dit programma gebruikt het uitwisselingsbestand van voorbeeld 1.
c Er worden 4 records teruggelezen en wel de eerste, een op basis
van
c de sleutelwaarden, daarna een met "NEXT" record en ten slotte de
c laatste.
c
      integer         error, j
      character*6     rdtype(4)
      character*20    uwbnam
      character*20    status

c Bij deze declaratie van variabelen is rekening gehouden met
c de veld (attribuut) definitie op het bilaterale bestand.
c De sleutelwaarde voor zoeken dient alle sleutels te bevatten,
c vandaar de characterlengte 32
c
      character*8     sv1
      character*24    sv2
      character*32    sltwrd
      character*24    cv1
```

```
      character*1     cv2
      character*256   cv3
      integer         iv1,iv2,iv3,iv4
      real            rv1,rv2,rv3,rv4,rv5
      double precision dv1,dv2,dv3

c Openen van file voor uitvoer van meldingen

      open (unit=9,file='voorb2.txt')

c Geef aantal initiele waarden, waaronder de filenamen
c Er wordt gelezen van uitwisselingsbestand van voorbeeld 1

      uwbnam = 'voorb1'
      status = 'READ'
      error  = 0
      sltwrd = ' '

      call SWALLOC (error)
      if (error .ne. 0) goto 9000

      call OPNUWB( error, uwbnam, status)
      if (error .ne. 0) goto 9000

c Vul een hulp-array voor lezen van records met de variable rdtype.
c
      rdtype(1) = 'FIRST'
      rdtype(2) = 'SEARCH'
      rdtype(3) = 'NEXT'
      rdtype(4) = 'LAST'

c Ga nu in een lus van 1 t/m 4 steeds een record lezen.
c De manier van record-lezen wordt bepaald door rdtype(j).
c
      do j = 1, 4

         if ( j .eq. 2 ) then
c           met 'SEARCH' , dus ook sleutelwaarde opgeven,
c           zie ook voorbeeld 1 voor sleutelwaarden en kies
c           hier waarde 255 in de sleutel
c
            write (sltwrd(1:8 ),'(i8.8)') 255
            write (sltwrd(9:32),'(i24.24)') 255
         else
c           de variabele sltwrd hoeft geen waarde te hebben bij
c           'FIRST' , 'NEXT' en 'LAST'
         endif

c        Lees een record
c
         call RDREC( error, uwbnam, 'ENSCIRD', rdtype(j), sltwrd)
         if (error .ne. 0) goto 9000

c        Lees alle velden uit het record
c
         call RDVLD_sc( error, uwbnam, 'ENSCIRD', 'sv1' ,sv1)
         if (error .ne. 0) goto 9000
         call RDVLD_sc( error, uwbnam, 'ENSCIRD', 'sv2' ,sv2)
         if (error .ne. 0) goto 9000
         call RDVLD_sc( error, uwbnam, 'ENSCIRD', 'cv1' ,cv1)
         if (error .ne. 0) goto 9000
         call RDVLD_sc( error, uwbnam, 'ENSCIRD', 'cv2' ,cv2)
         if (error .ne. 0) goto 9000
         call RDVLD_sc( error, uwbnam, 'ENSCIRD', 'cv3' ,cv3)
```

```
            if (error .ne. 0) goto 9000
            call RDVLD_i ( error, uwbnam, 'ENSCIRD', 'iv1' ,iv1)
            if (error .ne. 0) goto 9000
            call RDVLD_i ( error, uwbnam, 'ENSCIRD', 'iv2' ,iv2)
            if (error .ne. 0) goto 9000
            call RDVLD_i ( error, uwbnam, 'ENSCIRD', 'iv3' ,iv3)
            if (error .ne. 0) goto 9000
            call RDVLD_i ( error, uwbnam, 'ENSCIRD', 'iv4' ,iv4)
            if (error .ne. 0) goto 9000
            call RDVLD_r ( error, uwbnam, 'ENSCIRD', 'rv1' ,rv1)
            if (error .ne. 0) goto 9000
            call RDVLD_r ( error, uwbnam, 'ENSCIRD', 'rv2' ,rv2)
            if (error .ne. 0) goto 9000
            call RDVLD_r ( error, uwbnam, 'ENSCIRD', 'rv3' ,rv3)
            if (error .ne. 0) goto 9000
            call RDVLD_r ( error, uwbnam, 'ENSCIRD', 'rv4' ,rv4)
            if (error .ne. 0) goto 9000
            call RDVLD_r ( error, uwbnam, 'ENSCIRD', 'rv5' ,rv5)
            if (error .ne. 0) goto 9000
            call RDVLD_d ( error, uwbnam, 'ENSCIRD', 'dv1' ,dv1)
            if (error .ne. 0) goto 9000
            call RDVLD_d ( error, uwbnam, 'ENSCIRD', 'dv2' ,dv2)
            if (error .ne. 0) goto 9000
            call RDVLD_d ( error, uwbnam, 'ENSCIRD', 'dv3' ,dv3)
            if (error .ne. 0) goto 9000

c           Schrijf resultaten naar een uitvoer file
c
            write (9,'(2a, i4)') rdtype(j) , ' j=' , j
            write (9,'(a,a)') 'sv1=',sv1
            write (9,'(a,a24)') 'sv2=',sv2
            write (9,'(a,a24)') 'cv1=',cv1
            write (9,'(a,a1)') 'cv2=',cv2
            write (9,'(a,a64)') 'cv3=',cv3(1:64)
            write (9,'(a,i10)') 'iv1=',iv1
            write (9,'(a,i10)') 'iv2=',iv2
            write (9,'(a,i10)') 'iv3=',iv3
            write (9,'(a,i10)') 'iv4=',iv4
            write (9,'(a,f10.3)') 'rv1=',rv1
            write (9,'(a,f10.3)') 'rv2=',rv2
            write (9,'(a,f10.3)') 'rv3=',rv3
            write (9,'(a,f10.3)') 'rv4=',rv4
            write (9,'(a,f10.3)') 'rv5=',rv5
            write (9,'(a,e10.3)') 'dv1=',dv1
            write (9,'(a,e10.3)') 'dv2=',dv2
            write (9,'(a,e10.3)') 'dv3=',dv3
         enddo

 9000 continue
c
c Sluit het uitwisselingsbestand, ook in geval van een fout.
c Schrijf eerst het foutnr (of errror = 0 van de laatste correcte
c actie).
c
      write (9,'(a,i6)') 'Error = ', error
      call CLSUWB( error, uwbnam)
      write (9,'(a,i6)') 'Einde Voorbeeld 2: error = ', error

      close (9)

      stop

      end
```

7.3 Schijven van waardenreeksen

```
      program voorb_3

c In dit programma worden diverse waardenreeksen gecre-eerd op basis
van
c de waardenreeksdefinities die zijn gegeven in het bilaterale
bestand.
c
      integer         error, dim
      character*20    uwbnam
      character*20    uwfnam
      character*20    wrddef
      character*20    wrdnam
      character*20    status
      integer         i, j

c De declaratie van deze variabelen hangt nauw samen met de
definitie
c van de waardenreeksen en hun attributen op het bilaterale bestand.
c
      character*9     datum(20)
      integer         tijd(20)
      real            waarde(20)
      real            punt(2,30)
      real            x(20,15), y(20,15)


c Openen van file voor uitvoer van meldingen
c
      open (unit=9, file='voorb3.txt')

c Geef aantal initiele waarden, waaronder de filenamen
c
      uwbnam = 'voorb3'
      status = 'OVERWRITE'
      error  = 0
      uwfnam = 'test.uwf'

c Onderstaand de stappen, zoals ze ook in de handleiding zijn
c beschreven.

      write (9,'(a)' ) 'Start voorbeeld 3'

      call SWALLOC (error)
      if ( error .ne. 0 ) goto 9000

      call OPNUWB( error, uwbnam, status)
      if (error .ne. 0) goto 9000

c Cre-eer meerdere waardenreeksen. Waardenreeks definities staan op
c het uitwisselingsformaat bestand, dus ook op het bilateraal
stuurbestand.
c De waardenreeks krijgt soms een vaste lengte (die ten minste groot
c genoeg is) en soms lengte van 0 (aantal te schrijven waarden is
onbe-
c kend en wordt gegeven bij schrijven attributen)
c Lengte loopt via via variabele  dim
c
      dim    = 0
      wrddef = 'tijdreeks'
      wrdnam = 't1'
      call CREWRD( error, uwbnam, wrddef, wrdnam, uwfnam, dim )
```

```
      if (error .ne. 0) goto 9000

      dim    = 10
      wrddef = 'tijdreeks'
      wrdnam = 't2'
      call CREWRD( error, uwbnam, wrddef, wrdnam, uwfnam, dim )
      if (error .ne. 0) goto 9000

      dim    = 20
      wrddef = 'punt'
      wrdnam = 'p1'
      call CREWRD( error, uwbnam, wrddef, wrdnam, uwfnam, dim )
      if (error .ne. 0) goto 9000

      dim    = 1
      wrddef = 'rooster'
      wrdnam = 'r1'
      call CREWRD( error, uwbnam, wrddef, wrdnam, uwfnam, dim )
      if (error .ne. 0) goto 9000

c Vul de attributen van de waardenreeksen voor type tijdreeks en
schrijf ze weg.
c In het voorbeeld zijn het dummy waarden. Hoe de datum wordt
opgegeven ziet u
c hieronder. De tijd wordt in een integer van 6 cijfers opgegeven
(hhmmss)
c
      wrdnam = 't1'
      dim    = 15
      do j = 1, dim
         datum(j) (1: ) = '25mrt1999'
         tijd(j)        = (j-1) * 10000
         waarde(j)      = j
      enddo

      call WRWRD_c( error, uwbnam, wrdnam, 'DATUM',  dim, datum )
      if (error .ne. 0) goto 9000
      call WRWRD_i( error, uwbnam, wrdnam, 'TIJD',   dim, tijd )
      if (error .ne. 0) goto 9000
      call WRWRD_r( error, uwbnam, wrdnam, 'WAARDE', dim, waarde )
      if (error .ne. 0) goto 9000

c Nu waardereeks t2 van type tijdreeks
c
      wrdnam = 't2'
      dim    = 10
      do j = 1, dim
         datum(j) (1: ) = '26mrt1999'
         tijd(j)        = (j-1) * 10000 + 3000
         waarde(j)      = j * 2.0
      enddo

      call WRWRD_c( error, uwbnam, wrdnam, 'DATUM',  dim, datum )
      if (error .ne. 0) goto 9000
      call WRWRD_i( error, uwbnam, wrdnam, 'TIJD',   dim, tijd )
      if (error .ne. 0) goto 9000
      call WRWRD_r( error, uwbnam, wrdnam, 'WAARDE', dim, waarde )
      if (error .ne. 0) goto 9000

c Vul waardenreeks voor Punt en schrijf hem weg
c Waardenreeks-attribuut heeft afmeting 2 (voor x en y) en de reeks
c heeft lengte van 20 (dat kan hoewel er 30 bij declaratie hierboven
c staat)
c
```

```
      wrdnam = 'p1'
      dim    = 20
      do j = 1, dim
         punt(1,j) = j          ! x-coordinaat
         punt(2,j) = j+1        ! y-coordninaat
      enddo

      call WRWRD_r( error, uwbnam, wrdnam, 'COORD',  dim, punt )
      if (error .ne. 0) goto 9000

c Er is een waardenreeks r1 voor roosterpunten, gebaseerd op de
definitie
c van rooster in het bilaterale bestand.
c
c Voor het rooster r1 , maak de roosterpunten:
c Dimensie (= 'lengte' van rooster) is 1, we schrijven nl. 1 rooster
c voor x- en 1 rooster voor y-coordinaten weg.
c Coordinaten in het rooster: x=0...19, y=0...14
c LET OP: de afmetingen (attribuutdimensies) van x-coordinaten en
c         y-coordinaten zijn exact zoals bij bilateraal bestand
c
      dim    = 1
      wrdnam = 'r1'
      do j = 1, 15
         do i = 1, 20
            x(i,j) = i-1
            y(i,j) = j-1
         enddo
      enddo

      call WRWRD_r( error, uwbnam, wrdnam, 'X_COORD',  dim, x )
      if (error .ne. 0) goto 9000
      call WRWRD_r( error, uwbnam, wrdnam, 'Y_COORD',  dim, y )
      if (error .ne. 0) goto 9000


 9000 continue
c
c Sluit het uitwisselingsbestand, ook in geval van een fout.
c Schrijf eerst het foutnr (of errror = 0 van de laatste correcte
c actie).
c
      write (9,'(a,i6)') 'Error = ', error
      call CLSUWB( error, uwbnam)
      write (9,'(a,i6)') 'Einde Voorbeeld 3: error = ', error

      close (9)

      stop

      end
```

7.4 Lezen van waardenreeksen

```
      program voorb4

c In dit voorbeeld worden de waarden (attibuten) van waardenreeksen
uit voorbeeld 3
c teruggelezen van het uitwisselingsbestand.
c Ter controle zijn een aantal waarden naar een uitvoer file
geschreven.
```

```
c
      integer        error, dim
      character*20   uwbnam
      character*20   wrdnam
      character*20   status
      integer        i, j

c De declaratie van deze variabelen hangt nauw samen met de
definitie
c van de waardenreeksen en hun attributen op het bilaterale bestand.
c Er is ook voor gezorgd dat ze tenminste de minimale afmetingen
hebben
c om waarden terug te ontvangen bj de leesacties.
c
      character*9    datum(20)
      integer        tijd(20)
      real           waarde(20)
      real           punt(2,30)
      real           x(20,15), y(20,15)

c Openen van file voor uitvoer van meldingen
c
      open (9,file='voorb4.txt')

c Geef aantal initiele waarden, waaronder de filenamen.
c Uitwisselingsbestand met naam van voorbeeld 3.
c Omdat we alleen lezen is er geen bestand met uitwisselingsformaat
c nodig.
c
      uwbnam = 'voorb3'
      status = 'READ'
      error  = 0

c Nu de stappen voor het teruglezen.
c
      write (9,'(a)') 'Start Voorbeeld 4'

      call SWALLOC (error)
      if (error .ne. 0) goto 9000


      call OPNUWB( error, uwbnam, status)
      if (error .ne. 0) goto 9000

c Lees de twee tijdreeksen, dim geft de lengte van de tijdreeks aan
die
c we van file willen lezen.
c
      wrdnam = 't1'
      dim    = 15

      call RDWRD_c( error, uwbnam, wrdnam, 'DATUM',  dim, datum )
      if (error .ne. 0) goto 9000
      call RDWRD_i( error, uwbnam, wrdnam, 'TIJD',   dim, tijd )
      if (error .ne. 0) goto 9000
      call RDWRD_r( error, uwbnam, wrdnam, 'WAARDE', dim, waarde )
      if (error .ne. 0) goto 9000

c Schrjf alle waarden naar de uuitvoer file.
c
      do j = 1, dim
         write (9, '(A, I6.6, F8.2)' )datum(j) (1: ), tijd(j),
waarde(j)
      enddo
```

```
c Nu het lezen van de tweede tijdreeks. Daarvan worden de eerste en
de laatste
c waarde naar de uitvoer file geschreven.
c
      wrdnam = 't2'
      dim    = 10

      call RDWRD_c( error, uwbnam, wrdnam, 'DATUM',  dim, datum )
      if (error .ne. 0) goto 9000
      call RDWRD_i( error, uwbnam, wrdnam, 'TIJD',   dim, tijd )
      if (error .ne. 0) goto 9000
      call RDWRD_r( error, uwbnam, wrdnam, 'WAARDE', dim, waarde )
      if (error .ne. 0) goto 9000

      write (9, '(A, I6.6, F8.2)' )datum(1), tijd(1), waarde(1)
      write (9, '(A, I6.6, F8.2)' )datum(dim), tijd(dim),
waarde(dim)

c Lees waardenreeks voor Punt en schrijf een aantal waarden naar de
c uitvoer file.
c Waardenreeks-attribuut heeft afmeting 2 (voor x en y) en de reeks
c heeft lengte (variabele dim)  van 20 (dat kan hoewel er 30 bij
c de declaratie hierboven staat.

      wrdnam = 'p1'
      dim    = 20

      call RDWRD_r( error, uwbnam, wrdnam, 'COORD',  dim, punt )
      if (error .ne. 0) goto 9000

      write (9, '(4F8.2)' ) punt(1,1), punt(2,1), punt(1,5),
punt(2,5)
      write (9, '(4F8.2)' ) punt(1,18), punt(2,18),
     &                      punt(1,dim), punt(2,dim)

c Lees rooster ook terug.
c LET OP: de afmetingen (attribuutdimensies) van x-coordinaten en
c         y-coordinaten zijn exact zoals bij bilateraal bestand.
c         lengte van de waardenreeks is 1.
c
      dim    = 1
      wrdnam = 'r1'

      call RDWRD_r( error, uwbnam, wrdnam, 'X_COORD',  dim, x )
      if (error .ne. 0) goto 9000
      call RDWRD_r( error, uwbnam, wrdnam, 'Y_COORD',  dim, y )
      if (error .ne. 0) goto 9000

c Schrijf twee randen (de eerste x-coordinaat en de laatste y
coordinaat)
c naar de uitvoer file.
c
      do j = 1, 15
         write (9, '(F8.2, F8.2)' ) x(1, j) , y(1,j)
      enddo

      do i = 1, 20
         write (9, '(F8.2, F8.2)' ) x(i,15) , y(i,15)
      enddo


 9000 continue
```

```
c
c Sluit het uitwisselingsbestand, ook in geval van een fout.
c Schrijf eerst het foutnr (of errror = 0 van de laatste correcte
c actie).
c
      write (9,'(a,i6)') 'Error = ', error
      call CLSUWB( error, uwbnam)
      write (9,'(a,i6)') 'Einde Voorbeeld 4: error = ', error

      close (9)

      stop

      end
```

7.5 Opvraagfuncties

```
      program voorb_5

c Dit voorbeeld laat het gebruik van de inquire functies zien.
Daarbij
c worden de gegevens van tabellen en hun kolommen en waardenreeksen
en
c hun attributen naar een uitvoer file geschreven.
c
c
c Bij deze declaratie is er wel voor gezorgd dat de diverse
variabelen en
c arrays, waarin de informatie van de inquire routines terug komt
voldoende
c groot zijn als ook de karakterlenge.
c
      integer         error
      character*20    uwbnam
      character*20    status
      character*16    lsttbl(10), lstwrd(10)
      integer         nr_tbl, nr_wrd
      character*8     lstvld(30)
      character*1     lsttyp(30)
      integer         lstvnr(30)
      integer         nr_vld
      integer         nr_rec
      integer         attrdim(4)
      character*8     lstattr(30)
      integer         lstanr(30)
      integer         nr_rks, nr_attr
      character*16    wrddef
      character*1     bil
      integer         i, j, k


c Openen van file voor uitvoer van meldingen
c
      open (9,file='voorb5.txt')

c De uitwisselingsbestanden van voorbeeld 1 (tabellen) en
c voorbeeld 3 (waardenreeksen) worden gebruikt.
c We hebben dat in een lus van k = 1, 2 gedaan.

      status = 'READ'
      error  = 0

      do k = 1, 2
```

```
            if (k .eq. 1) then
               uwbnam = 'voorb1'
               write (9,'(a)' ) 'Start voorbeeld 5, tabellen'
            else
              uwbnam = 'voorb3'
              write (9,'(a)' ) 'Start voorbeeld 5, waardenreeksen'
            endif

c           Slechts eenmaal allocatie in een programma!!
c
            if ( k .eq. 1 ) then
               call SWALLOC (error)
               if ( error .ne. 0 ) goto 9000
            endif

            call OPNUWB( error, uwbnam, status)
            if (error .ne. 0) goto 9000

c           Geef aan dat er ruimte is (hier 10) voor de diverse
c           arrays voor de lijsten
c
            nr_tbl = 10
            nr_wrd = 10
            call inquwb( error, uwbnam, nr_tbl, lsttbl, nr_wrd, lstwrd)
            if ( error .ne. 0 ) goto 9000

            write (9, '(a,i6,a,i6)' ) 'nr_tbl =' , nr_tbl ,
     &                                ' nr_wrd =' , nr_wrd

c           Een loop voor alle tabellen van het uitwisselingsbestand.
Als
c           er geen tabellen zijn geldt nr_tbl=0
c
            do j = 1, nr_tbl

               write(9, '(a,a)' ) lsttbl(j)

c              Geef de ruimte voor de arrays lst... met informatie
c              over de velden
c              Na de aanroep is nr_vld het juiste aantal
c
               nr_vld = 30
               nr_rec = -1
               call inqtbl( error, uwbnam, lsttbl(j), nr_rec, bil,
     &                      nr_vld, lstvld, lsttyp, lstvnr )
               if ( error .ne. 0 ) goto 9000

               write(9, '(a,i8,a,a)' ) 'Aantal records =' , nr_rec ,
     &                                 ' bil = ' , bil

c              Schrijf de informatie van alle velden van een tabel naar
c              de uitvoer file.
c
               do i = 1, nr_vld
                  write(9, '(4x,a,a,i8)' ) lstvld(i), lsttyp(i),
lstvnr(i)
               enddo

            enddo   ! van de lus over de tabellen


c           Een loop voor alle waardenreeksen op het
uitwisselingsbestand.
```

```
c        Als er geen waardenreeksen zijn geldt nr_wrd=0
c
         do j = 1, nr_wrd

            write(9, '(a,a)' ) lstwrd(j)

c           Geef de ruimte voor de arrays lst... met informatie
c           over de attributen
c           Na de aanroep is nr_attr het juiste aantal
c
            nr_attr = 30
            nr_rks  = -1
            wrddef(1: ) = ' '
            call inqwrd( error, uwbnam, lstwrd(j), nr_rks, bil,
     &                   wrddef, attrdim,
     &                   nr_attr, lstattr, lsttyp, lstanr )
            if ( error .ne. 0 ) goto 9000

            write(9, '(a,i8,a,a,a,a)' ) 'Aantal waarden =' , nr_rks
,
     &                                  ' bil = ' , bil,
     &                                   ' Def = ' , wrddef
            write(9, '(a,4i6)' )  'Attr. dimensie ' ,  attrdim(1),
     &                            attrdim(2), attrdim(3),
attrdim(4)

c           Schrijf de informatie van alle attributen van een
waardenreeks
c           naar de uitvoer file.
c
            do i = 1, nr_attr
               write(9, '(4x,a,a,i8)' ) lstattr(i), lsttyp(i),
lstanr(i)
            enddo

         enddo   ! van de lus over de waardenreeksen

      enddo  ! van lus k = 1, 2 over de twee bestanden


 9000 continue
c
c Sluit het uitwisselingsbestand, ook in geval van een fout.
c Schrijf eerst het foutnr (of errror = 0 van de laatste correcte
c actie).
c Hier moet je rekening houden met het feit dat we met twee
uitwisselings-
c bestanden werken.
c
      write (9,'(a,i6,a,i3)') 'Error = ', error , 'bij k = ' , k

      do k = 1, 2
         if (k .eq. 1) then
           uwbnam = 'voorb1'
         else
           uwbnam = 'voorb3'
         endif

         call CLSUWB( error, uwbnam)
         write (9,'(a,i6)') 'Einde Voorbeeld 3: error = ', error

      enddo

      close (9)
```

```
stop

end
```

8 Beschrijving stuurbestanden

Dit hoofdstuk beschrijft het formaat van de stuurbestanden, zowel GW96 als bilateraal.

Beide stuurbestanden zijn als volgt opgebouwd:

- Het eerste deel bevat enige administratieve informatie;
- Het tweede deel bevat een beschrijving van alle gegevenselementen;
- Het derde deel bevat een beschrijving van alle entiteittypen.

De drie delen zijn gescheiden door een horizontale lijn met "-" tekens vanaf de eerste positie (kolom).

8.1 GW-(Adventus) classificatie

8.1.1 Deel 1: Administratieve gegevens

1e regel:

1. kolom 1: spatie
2. kolom 2 t/m 25: "Versie GW-classificatie "
3. kolom 26: spatie
4. kolom 27 t/m 90: De versie van de GW-classificatie

2e regel:

1. kolom 1: spatie
2. kolom 2 t/m 25: "Naam SDW-model "
3. kolom 26: spatie
4. kolom 27 t/m 90: (Hier wordt de naam van het SDW model automatisch ingevuld)

3e regel:

1. kolom 1: spatie
2. kolom 2 t/m 25: "Datum gegenereerd"
3. kolom 26: spatie
4. kolom 27 t/m 90: (Hier wordt de naam datum waarop het bestand is gegenereerd automatisch ingevuld)

4e regel:

1. kolom 1 t/m 90: "-" tekens

8.1.2 Deel 2: Gegevenselementen

5e regel en volgende (voor elk gegevens element één regel):

1. kolom 1: spatie
2. kolom 2 t/m 9: UvW-code van het gegevenselement
3. kolom 10: spatie
4. kolom 11 t/m 74: Gegevenselement (zoals gedefinieerd in GW-96)
5. kolom 75: spatie

6. kolom 76 t/m 87: Type (zoals gedefinieerd in GW'96: numeriek, alfanumeriek of datum)
7. kolom 88: spatie
8. kolom 89 t/m 96 De lengte van het gegevens element (zoals gedefinieerd in GW'96)
9. kolom 97: spatie
10. kolom 98 t/m 113: De eenheid van het gegevenselement (zoals opgenomen in GW'96). Dit veld wordt niet meegenomen definitiebestand.

Volgende regel:

1. kolom 1 t/m 90: "-" tekens

8.1.3 Deel 3: Entiteittypen

In dit blok worden 2 regelsoorten onderscheiden:

1. regels met de naam e.d. van het entiteittype en
2. regels met de gegevenselementen die behoren bij deze entiteittype. Bij deze gegevens elementen geldt dat gegevenselementen van een superentiteittype als eerste worden vermeld. De superentiteittypen worden ook afzonderlijk vermeld.

De regels met de naam van het entiteittype e.d. ziet er als volgt uit:

1. kolom 1: spatie
2. kolom 2 t/m 14: de UvW-code van het Adventus entiteittype
3. kolom 16: spaties
4. kolom 17 t/m 41: voor de entiteittypen staan hier spaties
5. kolom 42: spatie
6. kolom 43 t/m 132: aanvullende informatie over het entiteittyp of de lange GW naam van de entiteit. Deze info/naam wordt niet overgedragen naar de uitwisselingsfile of definitiefile.

Voor de attribuutregels geldt:

1. kolom 1 t/m 5: spaties
2. kolom 6 t/m 13: UvW-code van het gegevenselement
3. kolom 14: spatie
4. kolom 15: als het een sleutel betreft dan staat hier een "*" teken, anders een spatie. Elk entiteittype moet minimaal één sleutel hebben.
5. kolom 16: spatie
6. kolom 17 t/m 132: de naam van het gegevens element zoals opgenomen in GW'96. Deze naam wordt niet overgedragen naar de uitwisselingsfile of definitiefile.

De identificatie van de gegevenselementen en entiteittypen wordt geheel verzorgd aan de hand van de UvW-codes.

8.2 Bilaterale afspraken

8.2.1 Algemeen

In het stuurbestand met bilaterale afspraken kunnen gewijzigde definities worden opgegeven voor classificatie-gegevens die in GW'96 zijn gedefinieerd. Ook kunnen nieuwe gegevens worden gedefinieerd alsmede de waardenreeksen.

8.2.2 Deel 1: Administratieve gegevens

1e regel:

1. kolom 1: spatie
2. kolom 2 t/m 25: "Bilateraal stuurbestand "
3. kolom 26: spatie
4. kolom 27 t/m 90: Vrij te kiezen tekst als versie beschrijving.

2e regel:

1. kolom 1: spatie
2. kolom 2 t/m 25: "Toepassing "
3. kolom 26: spatie
4. kolom 27 t/m 90: (Hier kan worden opgegeven voor welke toepassing het stuurbestand is aangemaakt.)

3e regel:

1. kolom 1: spatie
2. kolom 2 t/m 25: "Datum aangemaakt"
3. kolom 26: spatie
4. kolom 27 t/m 90: (Hier moet de datum waarop het bestand is gegenereerd worden ingevuld)

4e regel:

1. kolom 1 t/m 90: "-" tekens

8.2.3 Deel 2: Gegevenselementen

5e regel en volgende (voor elk gegevens element één regel):

1. kolom 1: spatie
2. kolom 2 t/m 9: code van het gegevenselement, de zogenaamde bilateraal-code. Deze kan lijken op de systematiek van de UvW code
3. kolom 10: spatie
4. kolom 11 t/m 74: Gegevenselement (willekeurige naam voor het gegevens element)
5. kolom 75: spatie
6. kolom 76 t/m 87: Type (numeriek, alfanumeriek of datum)
7. kolom 88: spatie
8. kolom 89 t/m 96 De lengte van het gegevens element
9. kolom 97: spatie
10. kolom 98 t/m 113: Eventueel de eenheid van het gegevenselement. Dit veld wordt niet meegenomen in het definitiebestand

Volgende regel:

1. kolom 1 t/m 90: "-" tekens

8.2.4 Deel 3: Entiteittypen

In dit blok worden 2 regelsoorten onderscheiden:

1. regels met de naam e.d. van het entiteittype en
2. regels met de gegevenselementen die behoren bij deze entiteittype.

Een entiteittype kan ook een waardenreeks zijn. Zie hiervoor ook het voorbeeld van een bilateraal stuurbestand hierna in 8.4. De uitleg van waardenreeksen is gegeven in hoofdstuk 2.

De regels met de naam van het entiteittype ziet er als volgt uit:

1.	kolom 1:	spatie
2.	kolom 2 t/m 14:	De unieke code van het entiteittype, de zogenaamde bilaterale code. Deze kan lijken op de systematiek van de UvW code.
3.	kolom 16:	spatie
4.	kolom 17 t/m 41:	alleen als het een waardenreeks betreft, dan opsomming van de dimensies, gescheiden door een "," Bij entiteittypen is dit spaties.
5.	kolom 42:	spatie
6.	kolom 43 t/m 132:	een lange naam voor entiteittype of waardenreeks of een aanvullende beschrijving van het entiteittype / waardenreeks. Deze tekst wordt niet overgedragen naar het uitwisselingsformaat

Voor de attribuutregels geldt:

1.	kolom 1 t/m 5:	spaties
2.	kolom 6 t/m 13:	een bekende unieke code van het gegevenselement (de zogenaamde bilaterale code)
3.	kolom 14:	spatie
4.	kolom 15:	als het een sleutel betreft dan staat hier een "*"-teken, anders een spatie. Elk entiteittype, dat geen waardenreeks is, moet minimaal één sleutel hebben.
5.	kolom 16:	spatie
6.	kolom 17 t/m 132:	een willekeurige naam van het gegevens element Deze naam wordt niet overgedragen naar het uitwisselings-formaat

De identificatie van de gegevenselementen en entiteittypen wordt geheel verzorgd aan de hand van de bilaterale codes!

8.3 Voorbeeld GW96 stuurbestand

```
Versie GW-classificatie :1.0
Naam SDW model :         GW96
Datum gegenereerd :      01/05/1997
```

SNPANR	A-nummer natuurlijk persoon	Numeriek	10	
RAKAASTP	Aanduiding aktiviteit per stap per rechtsverhouding	Alfanumeriek	50	-
SADHUISA	Aanduiding bij huisnummer	Alfanumeriek	2	-
BAGNORMS	Aanduiding gehanteerde normeringssysteem	Alfanumeriek	50	-
SNPGEBRN	Aanduiding naamgebruik	Alfanumeriek	2	-
WPAREDOV	Aanduiding reden overschrijding	Alfanumeriek	50	-
ZRWVOORZ	Aanduiding van voorzieningsgebied rioolwaterzuiveringsinstallatie	Alfanumeriek	50	-
WPBVOLGO	Aanduiding volgorde procedurestap	Alfanumeriek	2	-
WOZWAARD	Aandeel waarde gebouwd	Numeriek	8	Dfl
ZBMVERHO	Aangesloten verhard oppervlak	Numeriek	6	Vierkant meters [m2]
ASBTYPE	Aanslag type	Numeriek	2	-
ASRBEDRG	Aanslagbedrag	Numeriek	8.2	gld
ASBBEDRG	Aanslagbiljet bedrag	Numeriek	16	guldens
ASBNUMMR	Aanslagnummer	Alfanumeriek	40	-
KPGAANSD	Aanslagpeil pomp gemaal (dag)	Alfanumeriek	6.3	m + NAP
KPGAANSN	Aanslagpeil pomp gemaal (nacht)	Alfanumeriek	6.3	m + NAP
HRIAANT	Aantal vervuilings equivalenten WVO	Numeriek	8.2	VE
ZBMAANTB	Aantal vervuilingseenheden bedrijven	Numeriek	8	VE
ZBMAANTO	Aantal vervuilingseenheden overige	Numeriek	8	VE
ZBMAANTR	Aantal vervuilingseenheden recreatie	Numeriek	0	VE
ZBMBEDR	Aantal aangesloten bedrijven	Numeriek	5	Aantal bedrijven
ZBMHUISH	Aantal aangesloten huishoudens	Numeriek	6	Aantal huishoudens
ZBMRECRE	Aantal aangesloten recreatiegebieden	Numeriek	5	Aantal recreatiegebieden
CLOUREWK	Aantal bedrijfsuren per week	Numeriek	3	Aantal uren per week
CLOUREJR	Aantal bedrijfsweken per jaar	Numeriek	2	Aantal weken per jaar
KBRAANTD	Aantal doorvaartopeningen	Numeriek	2	-
KDUBUIS	Aantal identieke duikerbuizen naast elkaar	Alfanumeriek	2	-
KSHAANPR	Aantal identieke palenrijen naast elkaar	Numeriek	2	-
KSLAANT	Aantal identieke sluizen naast elkaar	Numeriek	2	-
KSTAANT	Aantal identieke stuwen naast elkaar	Numeriek	2	-
VOVAANTA	Aantal identieke vastgoedelementen naast elkaar	Numeriek	3	-
ZBMINWEQ	Aantal inwoners equivalenten	Numeriek	8	-
KWLAANTK	Aantal kegels	Alfanumeriek	2	-

```
.....
.....
```

```
-------------------------------------------------------------------------------------------
ASB                                         Aanslagbiljet
    ASBIDENT * Id. aanslagbiljet
    ASBNUMMR   Aanslagnummer
    ASBVOLGN   Volgnummer aanslag
    SADGEMNA   Gemeentenaam
    ASBBEDRG   Aanslagbiljet bedrag
    ASBBELJR   Belastingjaar
    ASBTYPE    Aanslag type
    ASBDTMVA   Peildatum vanaf
    ASBDTMTM   Peildatum t/m
    ASBDTMDT   Dtm dagtekening
    ASBDTMV1   Vervaldatum 1e aanslagbiljet
    ASBDTMV2   Vervaldatum 2e aanslagbiljet
...........
...........
WAV                                         Aanvraag verg.
    WAVIDENT * Id. aanvraag verg.
    CLOIDENT   Id. loz.obj.
    TPBIDENT   Id. typeprocesbewaking
    WAVREDEN   Ind. reden aanvraag verg.
    WAVONTVA   Dtm ontvangst aanvraag verg.
    WAVONHER   Dtm verg. onherroepelijk
GAG                                         Administratief geb.
    GAGNAAM    Naam admin. geb.
    GAGSOORT   Srt admin. geb.
    GAGOPPVL   Opp. admin. geb.
    GAGBESTM   Ind. best. geb.
    GAGFUNC    Fnc.
    MPGKAART   Kaartbladaand. IGG/TNO
    GAGCBSNR   CBS nummer
    OPRIDENT * Id. gebied
    GEBSOORT   Srt gebied
SAD                                         Adres
    SADIDENT * Id. adres
    SADSTRAA   Straatnaam
    SADHUISN   Huisnummer
    SADHUISL   Huisletter
    SADHUIST   Huisnummertoevoeging
    SADHUISA   Aand. bij huisnummer
    SADPOSTB   Postbusnummer
    SADPOSTC   Postcode
    SADLOCAT   Loc.beschrijving
    SADPLATS   Woonplaatsnaam
    SADGEMNA   Gemeentenaam
    SADLAND    Landnaam
    SADPOSBU   Postcode buitenland
```

```
     SADBUURT   Buurtomschrijving
     SADWIJK    Wijkomschrijving
     SADDTMAG   Dtm aanvang geldigheid
     SADDTMEG   Dtm einde geldigheid
.....
.....
SNP                                  Natuurlijk persoon
     SNPANR     A-nummer natuurlijk persoon
     SNPSOFI    SoFi-nummer persoon
     SNPVRNM    Voornamen persoon
     SNPADEL    Adellijke titel/predikaat
     SNPACADT   Academische titel
     SNPVOORV   Voorvoegsels
     SNPVOORL   Voorletters
     SNPGNAAM   Geslachtsnaam
     SNPGEBD    Geboortedatum
     SNPOVLD    Overlijdensdatum
     SNPGESL    Geslachtsaand.
     SNPGEBRN   Aand. naamgebruik
     SNPHUWD    Dtm huwelijkssluiting
     SNPHUWO    Dtm huwelijksontbinding
     SUBIDENT • Id. subj.
     SUBAKR     Subj.nummer AKR
     SUBCODE    Subj.code
     SUBTELNR   Telefoonnummer
     SUBFAXNR   Telefaxnummer
     SUBNUMGI   Gironummer
     SUBCGIRO   Controlegetal gironummer
     SUBBNUM    Banknummer
     SUBGEHEI   Ind. geheim
     SUBTYPE    Srt subj.
.....................
.....................

Waardereeks001 *                     Dit is een 1 dimensionale waardereeksbeschrijving
     tijdstip   het tijdstip van de meting
     x          x-coordinaat van het meetpunt
     y          y-coordinaat van het meetpunt
     hoogte     de gemeten hoogte op locatie x,y
     wndsnlhd   de windsnelheid op locatie x,y
Waardereeks002   4,5,6,7,*           Dit is een 5 dimensionale waardereeksbeschrijving
     tijdstip   het tijdstip van de meting
     x          x-coordinaat van het meetpunt
     y          y-coordinaat van het meetpunt
     hoogte     de gemeten hoogte op locatie x,y
     wndsnlhd   de windsnelheid op locatie x,y
```

8.4 Voorbeeld bilateraal stuurbestand

Dit voorbeeld staat ook op de diskette van de Stekkerdoos Water onder de directory stuur.
Deze bilaterale file wordt ook gebruikt bij de voorbeelden (via het bestand met uitwisselingsformaat)

```
Bilateraal stuurbestand
Toepassing voor testen
Datum aangemaakt 24 maart 1999
-------------------------------------------------------------------------------------------
sv1      sleutel-veld1                                          alfanumeriek 8        eenheid van sv1
sv2      sleutel-veld2                                          alfanumeriek 24       eenheid van sv2
cv1      character-veld1                                        alfanumeriek 24       eenheid van cv1
cv2      character-veld2                                        alfanumeriek 1        eenheid van cv2
cv3      character-veld3                                        alfanumeriek 256      eenheid van cv3
iv1      integer-veld1                                          numeriek     4        eenheid van iv1
iv2      integer-veld2                                          numeriek     4        eenheid van iv2
iv3      integer-veld3                                          numeriek     4        eenheid van iv3
iv4      integer-veld4                                          numeriek     4        eenheid van iv4
rv1      real-veld1                                             numeriek     5.3      eenheid van rv1
rv2      real-veld2                                             numeriek     5.3      eenheid van rv2
rv3      real-veld3                                             numeriek     5.3      eenheid van rv3
rv4      real-veld4                                             numeriek     5.3      eenheid van rv4
rv5      real-veld5                                             numeriek     5.3      eenheid van rv5
dv1      double precision1-veld1                                numeriek     12.3     eenheid van dv1
dv2      double precision1-veld2                                numeriek     12.3     eenheid van dv2
dv3      double precision1-veld3                                numeriek     12.3     eenheid van dv3
datum    de datum als bijv. 03mrt1999                           alfanumeriek 9        --
tijd     tijdstip als bijv. hhmmss                              numeriek     4        --
waarde   real waarde                                            numeriek     6.3
coord    coordinaat (real waarde)                               numeriek     6.3      --
x_coord  x coordinaat (real waarde)                             numeriek     6.3      --
y_coord  y coordinaat (real waarde)                             numeriek     6.3      --
waterst  waterstand (real waarde)                               numeriek     6.3      --
-------------------------------------------------------------------------------------------
ENS                                  Entiteit met 2 sleutel-velden
    sv1      * sleutel-veld1
    sv2      * sleutel-veld2
ENSC                                 Entiteit met 2 sleutel-velden en 2 character-velden
    sv1      * sleutel-veld1
    sv2      * sleutel-veld2
```

```
    cv1         character-veld1
    cv2         character-veld2
ENSI                                  Entiteit met 2 sleutel-velden en 2 integer-velden
    sv1       * sleutel-veld1
    sv2       * sleutel-veld2
    iv1         integer-veld1
    iv2         integer-veld2
ENSR                                  Entiteit met 2 sleutel-velden en 2 real-velden
    sv1       * sleutel-veld1
    sv2       * sleutel-veld2
    rv1         real-veld1
    rv2         real-veld2
ENSD                                  Entiteit met 2 sleutel-velden en 2 double precision-velden
    sv1       * sleutel-veld1
    sv2       * sleutel-veld2
    dv1         double precision-veld1
    dv2         double precision-veld2
ENSCIRD                               Entiteit met 2 sleutel-veld en velden van elk datatype
    sv1       * sleutel-veld1
    sv2       * sleutel-veld2
    cv1         character-veld1
    cv2         character-veld2
    cv3         character-veld3
    iv1         integer-veld1
    iv2         integer-veld2
    iv3         integer-veld3
    iv4         integer-veld4
    rv1         real-veld1
    rv2         real-veld2
    rv3         real-veld3
    rv4         real-veld4
    rv5         real-veld5
    dv1         double precision-veld1
    dv2         double precision-veld2
    dv3         double precision-veld3
ABCDEFGHIJKLM                         Entiteitnaam 13 characters lang
    sv1       * sleutel-veld1
    cv1         character-veld1
TIJDREEKS       1                     waardenreeks entiteit
    datum       de datum als bijv. 03mrt1999
    tijd        tijdstip als bijv. hhmmss
    waarde      real waarde
```

```
PUNT          2                        waardenreeks entiteit
    coord     coordinaat (real waarde)
GRID          12, 25                   waardenreeks entiteteit
    coord     coordinaat (real waarde)
ROOSTER       20 , 15                  waardenreeks entiteit
    x_coord   x coordinaat (real waarde)
    y_coord   y coordinaat (real waarde)
BLOK          9, 8                     waardenreeks entiteit
    waterst   waterstand (real waarde)
```

9 Overzicht Nefis foutmeldingen

Deze foutmeldingen zijn overgenomen van de NEFIS handleiding. Een NEFIS fout betekent dat er een definitie-, lees- of schrijfopdracht op NEFIS niveau fout is gegaan.

Let op:
Veelal ontstaat een NEFIS fout door een foutieve naam van een entiteittype, attribuut of waardereeks. Ook kan de melding veroorzaakt worden door het afwezig, "read-only" zijn van de noodzakelijke bestanden. Ook een volle schijf geeft een NEFIS foutmelding. In alle gevallen is het waardevol om bij een NEFIS fout op deze punten een controle uit te voeren.

In het uiterste geval kan contact opgenomen worden met de beheersorganisatie.

-1011 CLSDAT: FILE NOT CLOSED
→ Data file (DATFDS) cannot be closed.

-1021 CLSDAT: HASHTABLE NOT WRITTEN TO FILE
→ The hash table located in the "DATFDS" parameter has not been written to the data file and the data file is not closed.

-2011 CLSDEF: FILE NOT CLOSED
→ Definition file (DEFFDS) cannot be closed.

-2021 CLSDEF: HASH TABEL NOT WRITTEN TO FILE
→ The hash table located in the "DEFFDS" parameter has not been written to the definition file and the file is not closed.

-3011 CREDAT: GROUP DEFINITION DOES NOT EXIST
→ The group definition name given (GRPDEF) is not in the definition file (DEFFDS). (Het .def bestand ontbreekt)

-3021 CREDAT: ERROR ON READING DEFINITION FILE
→ A read error occurred when the group definition was being read from the definition file (DEFFDS).

-3031 CREDAT: ERROR ON READING DATA FILE
→ A read error occurred when the group definition was being read from the data file (DATFDS).

-3042 CREDAT: DATA GROUP ALREADY EXISTS
→ There is already a data group with the name "GRPNAM" in the data file (DATFDS).

-3051 CREDAT: ERROR ON WRITING DATA FILE
→ An error occurred when writing to the data file (DATFDS).

-3061 CREDAT: ERROR ON WRITING DATA FILE
→ An error occurred when writing to the data file (DATFDS).

-3071 CREDAT: ERROR ON WRITING DATA FILE
→ An error occurred when writing to the data file (DATFDS).

-4011 DEFCEL: NUMBER OF ELEMENTS OUT OF LIMITS
→ The number of elements given (NELEMS) is out of limits (see user manual).

-4021 DEFCEL: ELEMENT DOES NOT EXIST
→ One of the elements given (ELMNMS) is not defined in the definition file (DEFFDS) or an error occurred on reading from the definition file.

-4031 DEFCEL: ERROR ON READING DEFINITION FILE
→ An error occurred when reading from the definition file (DEFFDS).

-4042 DEFCEL: CELL NAME ALREADY EXIST
→ There is already a cell definition with the name "CELNAM" in the definition file (DEFFDS).

-4051 DEFCEL: ERROR ON WRITING DEFINITION FILE
→ An error occurred on writing to the definition file (DEFFDS).

-4061 DEFCEL: ERROR ON WRITING DEFINITION FILE
→ An error occurred on writing to the definition file (DEFFDS).

-5011 DEFELM: ELEMENT TYPE UNKNOWN
→ The element type given (ELMTYP) is unknown (see user manual).

-5021 DEFELM: NUMBER OF DIMENSIONS OUT OF LIMITS
→ The number of dimensions given (ELMNDM) is outside the permitted limits (see user manual).

-5031 DEFELM: ERROR ON READING DEFINITION FILE
→ An error occurred on reading from the definition file (DEFFDS).

-5042 DEFELM: ELEMENT ALREADY DEFINED
→ There is already an element with the name "ELMNAM" in the definition file (DEFFDS).

-5051 DEFELM: ERROR ON WRITING DEFINITION FILE
→ An error occurred on writing to the definition file (DEFFDS).

-5061 DEFELM: ERROR ON WRITING DEFINITION FILE
→ An error occurred on writing to the definition file (DEFFDS).

-6011 DEFGRP: NUMBER OF DIMENSIONS OUT OF LIMITS
→ The number of dimensions given (GRPNDM) is outside the permitted limits (see user manual).

-6021 DEFGRP: CELL DOES NOT EXIST
→ The cell given (CELNAM) has not been found in the definition file (DEFFDS) or there was a read error on the definition file.

-6031 DEFGRP: ERROR ON READING DEFINITION FILE
→ An error occurred on reading from the definition file (DEFFDS).

-6042 DEFGRP: GROUP DEFINITION NAME ALREADY EXISTS
→ There is already a group definition with the name "GRPDEF" in the definition file (DEFFDS).

-6051 DEFGRP: ERROR ON WRITING DEFINITION FILE
→ An error occurred on writing to the definition file (DEFFDS).

-6061 DEFGRP: ERROR ON WRITING DEFINITION FILE
→ An error occurred on writing to the definition file (DEFFDS).

-7011 FLSDAT: HASH TABLE NOT WRITTEN TO FILE
→ The hash table located in the "DATFDS" parameter has not been written to the data file.

-8011 FLSDEF: HASH TABLE NOT WRITTEN TO FILE
→ The hash table located in the "DEFFDS" parameter has not been written to the definition file.

-9011 GETELM: DATA GROUP NAME DOES NOT EXIST
→ The data group name given (GRPNAM) is not in the data file (DATFDS).

-9021 GETELM: GROUP DEFINITION DOES NOT EXIST
→ The group definition name (GRPNAM) for the data group was not found in the definition file (DEFFDS).

-9031 GETELM: ERROR ON READING DEFINITION FILE
→ An error occurred on reading cell information from the definition file(DEFFDS).

-9041 GETELM: ERROR ON READING DEFINITION FILE
→ An error occurred on reading from the definition file (DEFFDS).

-9051 GETELM: BUFFER TOO SMALL
→ The buffer length given (BUFLEN) is too small to accept the data requested.

-9061 GETELM: ERROR ON READING DATA FILE
→ An error occurred on reading the data requested from the data file (DATFDS).

-9071 GETELM: FUNCTION IS USED FOR A GROUP WITH VARIABLE DIMENSION
→ The given group (GRPNAM) is a group with a variable dimension. Data in such a group cannot be read with the GETELM function but must be read with the GETELT function.

-10011 GETIAT: ERROR ON READING DATA FILE
→ An error occurred on reading the data requested from the data file (DATFDS).

-10021 GETIAT: DATA GROUP NAME DOES NOT EXIST
→ The data group name given (GRPNAM) was not found in the data file (DATFDS).

-10031 GETIAT: ERROR ON READING DATA FILE
→ An error occurred on reading attributes from the data file (DATFDS).

-10051 GETIAT: ATTRIBUTE DOES NOT EXIST
→ The required attribute (ATTNAM) is not in the data file

-11011 GETRAT: ERROR ON READING DATA FILE
→ An error occurred on reading from the data file (DATFDS).

-11021 GETRAT: DATA GROUP NAME DOES NOT EXIST
→ The data group name given (GRPNAM) was not found in the data file (DATFDS).

-11031 GETRAT: ERROR ON READING DATA FILE→ An error occurred on reading attributes from the data file (DATFDS).

-11051 GETRAT: REQUIRED ATTRIBUTE DOES NOT EXIST→ The required attribute (ATTNAM) is not in the data file

-12011 GETSAT: ERROR ON READING DATA FILE→ An error occurred on reading from the data file (DATFDS).

-12021 GETSAT: DATA GROUP NAME DOES NOT EXIST→ The data group name given (GRPNAM) was not found in the data file (DATFDS).

-12031 GETSAT: ERROR ON READING DATA FILE→ An error occurred on reading attributes from the data file (DATFDS).

-12051 GETSAT: REQUESTED ATTRIBUTE DOES NOT EXIST
→ The attribute requested (ATTNAM) is not in the data file.

-13011 INQCEL: ERROR ON READING DEFINITION FILE
→ An error occurred on reading from the definition file (DEFFDS).

-13021 INQCEL: CELL NAME DOES NOT EXIST
→ The cell given (CELNAM) is not in the definition file (DEFFDS).

-13031 INQCEL: ERROR ON READING DEFINITION FILE
→ An error occurred on reading data from the given cell (CELNAM).

-13041 INQCEL: ERROR ON READING DEFINITION FILE
→ An error occurred on reading data from the given cell (CELNAM).

-13052 INQCEL: TOO MANY ELEMENTS IN CELL
→ The cell requested has more elements than can be contained in the given array (ELMNMS (1:NELEMS)).

-14011 INQDAT: ERROR ON READING DATA FILE → An error occurred on reading the data group name from the data file (DATFDS).

-14022 INQDAT: DATA GROUP NAME DOES NOT EXIST → The data group name given (GRPNAM) was not found in the data file (DATFDS).

-14031 INQDAT: ERROR ON READING DATA FILE → An error occurred on reading data group information from the data file (DATFDS).

-15011 INQELM: ERROR ON READING DEFINITION FILE → An error occurred on reading the element name (ELMNAM) from the definition file (DEFFDS).

-15022 INQELM: ELEMENT DOES NOT EXIST → The element given (ELMNAM) was not found in the definition file (DEFFDS).

-15031 INQELM: ERROR ON READING DEFINITION DEFINITION FILE → A read error occurred on the definition file (DEFFDS) when reading information relating to the element given (ELMNAM).

-16011 INQFST: ERROR ON READING DATA FILE
→ An error occurred on reading from the data file (DATFDS)

-16021 INQFST: READ ERROR OR END OF FILE
→ When reading from the data file (DATFDS) an error occurred or the end of the file was reached.

-17011 INQNXT: ERROR ON READING DATA FILE
→ When reading from the data file (DATFDS) an error occurred.

-17021 INQNXT: READING ERROR OR END OF FILE → When reading from the data file (DATFDS) an error occurred or the end of the file was reached.

-18011 INQGRP: ERROR ON READING DEFINITION FILE → A read error occurred on the definition file (DEFFDS) when reading the group definition given (GRPDEF).

-18022 INQGRP: GROUP DEFINITION DOES NOT EXIST → The group definition name given (GRPDEF) is not in the definition file (DEFFDS).

-18031 INQGRP: ERROR ON READING DEFINITION FILE → An error occurred on the definition file (DEFFDS) when reading information relating to the group definition of GRPDEF.

-19011 OPNDAT: ERROR ON OPENING DATA FILE → An error occurred on opening the specified data file (DATFNM). It is likely that "DATFNM" is not a correct file name.

-20011 OPNDEF: ERROR ON OPENING DEFINITION FILE → An error occurred on opening the specified definition file (DEFFNM). It is likely that "DEFFNM" is not a correct file name.

-21011 PUTELM: DATA GROUP DOES NOT EXIST OR READ ERROR → The data group name given (GRPNAM) is not in the data file or a read error has occurred.

-21021 PUTELM: GROUP DEFINITION DOES NOT EXIST → The group definition for the data group name (GRPNAM) is not in the definition file (DEFFDS).

-21031 PUTELM: ERROR ON READING DEFINITION → An error occurred on reading group definition information from the definition file (DEFFDS).

-21041 PUTELM: ELEMENT DOES NOT EXIST → The element given (ELMNAM) was not found in the definition of the data group.

-21051 PUTELM: ERROR ON WRITING DATA FILE → An error occurred on writing data to the data file (DATFDS).

-21061 PUTELM: FUNCTION WAS USED FOR A GROUP WITH VARIABLE DIMENSION
→ The given group (GRPNAM) is a group with a variable dimension. Data in such a group cannot be read with the PUTELM function but must be read with the PUTELT function.

-22011 PUTIAT: ERROR ON READING DATA FILE
→ An error occurred on reading from the data file (DATFDS). Probably the file is READ only.

-22021 PUTIAT: DATA GROUP DOES NOT EXIST
→ The data group name given (GRPNAM) is not in the data file (DATFDS) or a read error has occurred.

-22031 PUTIAT: ERROR ON READING DATA FILE
→ An error occurred on reading from the data file (DATFDS).

-22041 PUTIAT: ERROR ON WRITING DATA FILE
→ An error occurred on writing to the data file (DATFDS).

-22051 PUTIAT: NO MORE SPACE FOR ATTRIBUTE
→ The number of attributes in this data group (GRPNAM) has reached the maximum limit (see user manual).

-23011 PUTRAT: ERROR ON READING DATA FILE
→ An error occurred on reading from the data file (DATFDS).

-23021 PUTRAT: DATA GROUP DOES NOT EXIST
→ The data group name given (GRPNAM) is not in the data file (DATFDS) or a read error has occurred.

-23031 PUTRAT: ERROR ON READING DATA FILE
→ An error occurred on reading from the data file (DATFDS).

-23041 PUTRAT: ERROR ON WRITING DATA FILE
→ An error occurred on writing to the data file (DATFDS).

-23051 PUTRAT: NO MORE SPACE FOR ATTRIBUTE
→ The number of attributes in this data group (GRPNAM) has reached the maximum limit (see user manual).

-24011 PUTSAT: ERROR ON READING DATA FILE
→ An error occurred on reading from the data file (DATFDS).

-24021 PUTSAT: DATA GROUP DOES NOT EXIST
→ The data group name given (GRPNAM) is not in the data file (DATFDS) or a read error has occurred.

-24031 PUTSAT: ERROR ON READING DATA FILE
→ An error occurred on reading from the data file (DATFDS).

-24041 PUTSAT: ERROR ON WRITING DATA FILE
→ An error occurred on writing to the data file (DATFDS).

-24051 PUTSAT: NO MORE SPACE FOR ATTRIBUTE
→ The number of attributes in this data group (GRPNAM) has reached the maximum limit (see user manual).

-25011 GETELT: DATA GROUP NAME DOES NOT EXIST → The given data group name (GRPNAM) is not in the the data file (DATFDS).

-25021 GETELT: GROUP DEFINITION DOES NOT EXIST → Group definition name (GRPNAM) corresponding to the data group was not found in the definition file (DEFFDS).

-25031 GETELT: ERROR ON READING DEFINITION FILE → On reading cell information from the definition file (DEFFDS) an error occurred.

-25041 GETELT: ERROR ON READING DEFINITION FILE → On reading element information from the definition file (DEFFDS) an error occurred.

-25051 GETELT: BUFFER TOO SMALL → Given buffer size (BUFLEN) is too small to store the required data.

-25061 GETELT: INDEX DOES NOT EXIST → Data which corresponds to one or more of the requested indices of the variable dimension does not exist.

-25071 GETELT: ERROR ON READING DATA FILE → On reading the requested data from the data file (DATFDS) an error occurred.

-25081 GETELT: ERROR IN GIVEN INDICES → One of the start indices is larger than the corresponding end value.

-25091 GETELT: ERROR IN GIVEN INCREMENT → One of the given increments is smaller than 0 or equal.

-25101 GETELT: ERROR IN GIVEN START INDEX → One of the given start values is smaller than 0 or equal.

-25111 GETELT: ERROR IN GIVEN END INDEX → One of the given end values is larger than the maximum size (which obviously does not hold for the variable dimension).

-26011 PUTELT: DATA GROUP NAME DOES NOT EXIST → The given data group name (GRPNAM) is not in the data file (DATFDS).

-26021 PUTELT: GROUP DEFINITION DOES NOT EXIST →Group definition name corresponding to the data group (GRPNAM) was not found in the definition file (DEFFDS).

-26031 PUTELT: ERROR ON READING DEFINITION FILE → On reading cell information from the definition file (DEFFDS) an error occurred.

-26041 PUTELT: ERROR ON READING DEFINITION FILE → On reading element information from the definition file (DEFFDS) an error occurred).

-26051 PUTELT: ON WRITING THE DATA FILE AN ERROR OCCURRED →
An error occurred on writing the data file.

-26061 PUTELT: ON WRITING THE DATA FILE AN ERROR OCCURRED →
An error occurred on writing the data file.

-26081 PUTELT: ERROR IN GIVEN INDICES
→ One of the start indices is larger than the corresponding end value.

-26091 PUTELT: ERROR IN GIVEN INCREMENT
→ One of the given increments is smaller than 0 or equal.

-26101 PUTELT: ERROR IN GIVEN START INDEX
→ One of the given start values is smaller than 0 or equal.

-26111 PUTELT: ERROR IN GIVEN END INDEX
→ One of the given end values is larger than the maximum value. (Obviously this does not concern the variable dimension).

-27012 GETHDF: HEADER TOO SMALL
→ The buffer for the header which was given by the user is too small. This buffer must have a size of at least 60 characters.

-27022 GETHDF: ON READING THE DEFINITION FILE AN ERROR OCCURRED
→ On reading the header from the definition file a read error occurred.

-28012 GETHDT: HEADER TOO SMALL
→ The buffer for the header given by the user is too small. This buffer must at least have a size of 60 characters.

-28022 GETHDT: ON READING THE DATA FILE AN ERROR OCCURRED
→ On reading the header from the data file a read error occurred.

-29011 INQFIA: ON READING THE DATA FILE AN ERROR OCCURRED
→ On reading data from the data file (DATFDS) an error occurred.

-29021 INQFIA: DATA GROUP DOES NOT EXIST
→ The given data group (GRPNAM) is not in the data file.

-29031 INQFIA: ON READING THE DATA FILE AN ERROR OCCURRED
→ On reading data from the data file (DATFDS) an error occurred.

-29051 INQFIA: DATA GROUP DOES NOT HAVE ATTRIBUTES
→ The given data group (GRPNAM) does not have attributes.

-30051 INQNIA: DATA GROUP DOES NOT HAVE ATTRIBUTES
→ The given data group (GRPNAM) does not have attributes.

-31011 INQFRA: ON READING THE DATA FILE AN ERROR OCCURRED
→ On reading data from the data file (DATFDS) an error occurred.

-31021 INQFRA: DATA GROUP DOES NOT EXIST
→ The given data group (GRPNAM) is not in the data file.

-31031 INQFRA: ON READING THE DATA FILE AN ERROR OCCURRED
→ On reading data from the data file (DATFDS) an error occurred.

-31051 INQFRA: DATA GROUP DOES NOT HAVE ATTRIBUTES
→ The given data group (GRPNAM) does not have attributes.

-32051 INQNRA: DATA GROUP DOES NOT HAVE ATTRIBUTES
→ The given data group (GRPNAM) does not have attributes.

-33011 INQFSA: ON READING THE DATA FILE AN ERROR OCCURRED
→ On reading data from the data file (DATFDS) an error occurred.

-33021 INQFSA: DATA GROUP DOES NOT EXIXST
→ The given data group (GRPNAM) is not in the data file.

-33031 INQFSA: ON READING THE DATA FILE AN ERROR OCCURRED
→ On reading data from the data file (DATFDS) an error occurred.

-33051 INQFSA: DATA GROUP DOES NOT HAVE ATTRIBUTES
→ The given data group (GRPNAM) does not have attributes.

-34051 INQNSA: DATA GROUP DOES NOT HAVE ATTRIBUTES
The given data group (GRPNAM) does not have attributes.

Stekkerdoos Water (versie 3.0)

Functioneel Ontwerp

Inhoud

1 Inleiding **1**

2 De omgeving van het systeem **2**

2.1 ..... Doelstelling 2

2.2 ..... Gebruikte symbolen 3

2.3 ..... Context diagram 4

2.3.1 Elementen in Context diagram 5
2.3.2 Gegevensstromen in Context diagram 5
2.3.3 Buffers in Context diagram 5
2.3.4 Gebeurtenissenlijst 6
2.3.5 De afbakening 7

3 Het gedrag van het systeem **8**

3.1 ..... Inleiding 8

3.2 ..... Uitwisselingsformaat generator 9

3.2.1 Dataflow diagram 9
3.2.2 Functies 10

3.3 ..... De stekkerdoos 11

3.3.1 Context diagram 11
3.3.2 Begrippen 12
3.3.3 Waardenreeksen in de Stekkerdoos 14
3.3.4 Het idee 18
3.3.5 Functionaliteit van de stekkerdoos 19
3.3.6 Openen bestand 21
3.3.7 Definiëren tabel 24
3.3.8 Vullen veld 26
3.3.9 Schrijven record 27
3.3.10 Lezen record 29
3.3.11 Ophalen veld 31
3.3.12 Schrijven kolom of reeks 32
3.3.13 Lezen kolom of reeks 34
3.3.14 Opvragen gegevens 35
3.3.15 Sluiten bestand 36

Appendices

A ..... Additionele documenten

Figuren

Figuur 1 ..... Gebruikte symbolen 3
Figuur 2 ..... Context diagram 4
Figuur 3 ..... Uitwisselingsformaat generator 9
Figuur 4 ..... Context diagram (stekkerdoos) 11
Figuur 5 ..... Recordbuffer 18
Figuur 6 ..... Openen bestand 21
Figuur 7 ..... Openen nieuw bestand 22
Figuur 8 ..... Openen bestaand bestand 23
Figuur 9 ..... Definiëren tabel 24
Figuur 10 ..... Vullen veld 26
Figuur 11 ..... Schrijven record 27
Figuur 12 ..... Lezen record 29
Figuur 13 ..... Ophalen veld 31
Figuur 14 ..... Schrijven kolom 32
Figuur 15 ..... Lezen kolom 34
Figuur 16 ..... Opvragen gegevens 35
Figuur 17 ..... Sluiten bestand 36

1 Inleiding

Dit document is geschreven in het kader van de opdracht tot verbetering van de performance van Stekkerdoos Water.
Daar waar in de versies 1.0 en 2.0 uitgangspunt was dat de applicatie eenmalig (enkele keren) met het uitwisselingsbestand communiceerde, staat in deze versie 3.0 een continue koppeling tussen applicatie en uitwisselingsbestand centraal en daarmee wordt de performance belangrijk..
Omdat dit een geheel andere insteek is dan bij de vorige versies, is besloten een geheel nieuw ontwerp te maken, gebruik makend van de kennis van vorige versies.

In hoofdstuk 2 worden de grenzen van het systeem besproken met als doel te onderkennen welke onderdelen wel en welke onderdelen niet tot de Stekkerdoos Water behoren.

In hoofdstuk 3 wordt de functionaliteit besproken. Hier worden ook de begrippen besproken die ten grondslag liggen aan de functionaliteit.

2 De omgeving van het systeem

De eerste stap is afbakenen van de grenzen van het systeem; m.a.w. wat hoort nog wel en wat hoort niet tot het systeem.
Om daar inzicht in te krijgen is het omgevingsmodel (environmental model) opgesteld. Binnen het omgevingsmodel komen aan de orde:

- doel van het systeem (=doelstelling),
- de elementen waarmee het systeem gegevens uitwisselt en de bijbehorende gegevensstromen en buffers (=context diagram) en
- een opsomming van de gebeurtenissen binnen en op het systeem (=gebeurtenissen lijst).

2.1 Doelstelling

Het primaire doel is:

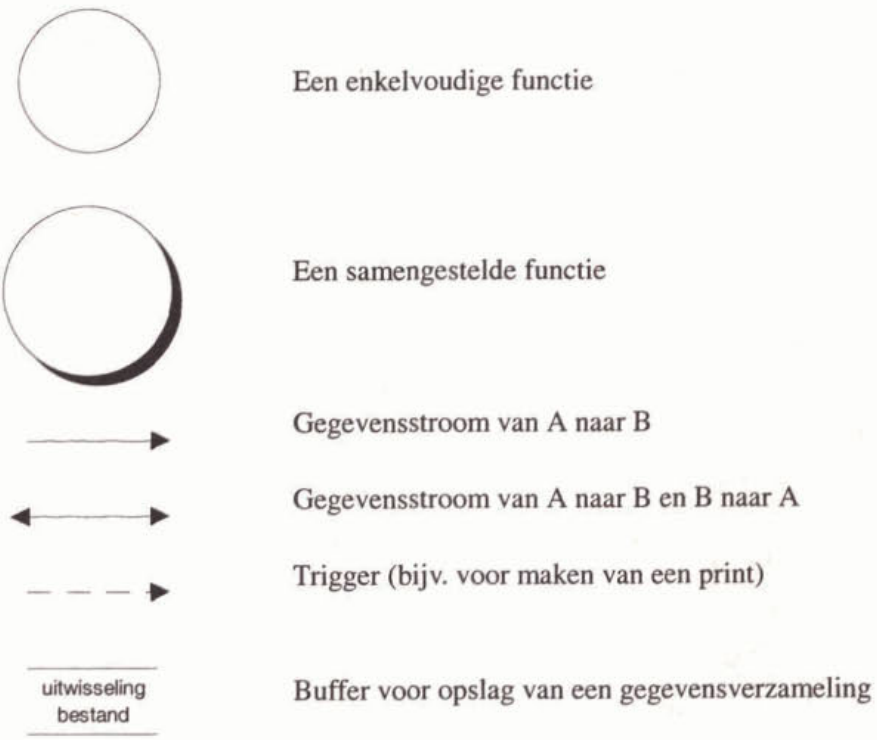
het kunnen uitwisselen van gegevens volgens het Adventus datamodel, eventueel aangevuld met aanvullende bilaterale afspraken, middels zogenoemde uitwisselingsbestanden.

Als randvoorwaarden hierbij gelden:

- er moeten meerdere uitwisselingsbestanden tegelijkertijd geschreven en gelezen kunnen worden.
- de uitwisselingsbestanden moeten platform-onafhankelijk zijn.
- voor het lezen en schrijven moet gebruik gemaakt worden van de Nefis-bibliotheek.
- de performance van het lezen en schrijven moet minstens een orde beter zijn dan de huidige performance; de exacte performance-eisen worden in het testplan vastgelegd.

2.2 Gebruikte symbolen

De volgende symbolen en bijbehorende betekenis worden gebruikt in de Context en Dataflow Diagrammen.



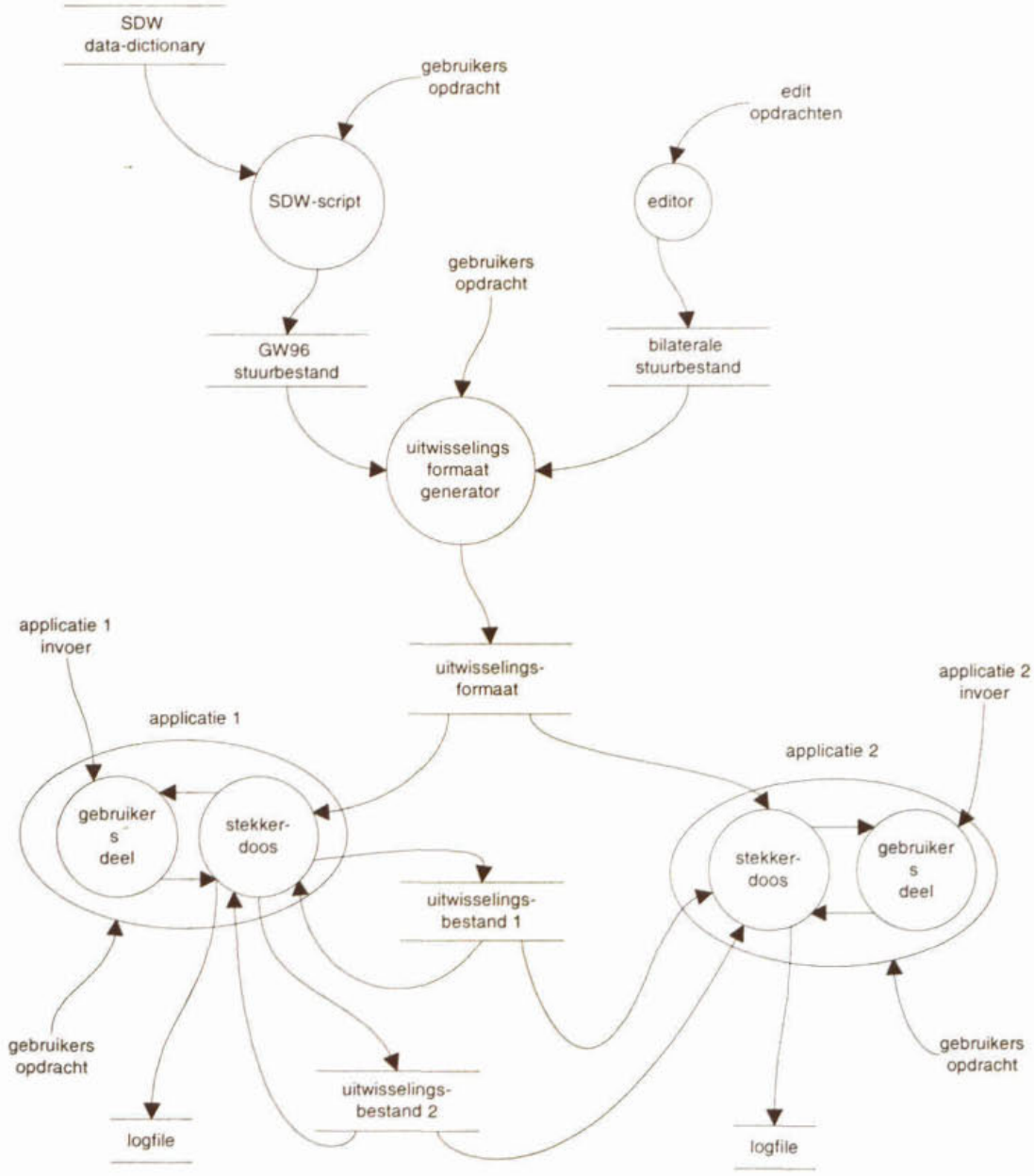
Figuur 1 Gebruikte symbolen

Nadere verklaring

Met een enkelvoudige functie wordt een proces bedoeld waarmee een bepaalde activiteit wordt uitgevoerd. Bijvoorbeeld: definiëren entiteit, schrijven record, lezen kolom, etc.

Samengestelde functies zullen worden opgesplitst in enkelvoudige functies.
Een enkelvoudig functie wil zeggen dat er in dit Functioneel Ontwerp geen verdere detaillering meer plaats zal vinden. Op basis van deze enkelvoudige functies kan worden vastgesteld WAT het systeem moet kunnen. In het Detail Ontwerp kunnen deze enkelvoudige functies verder worden uitgewerkt om inzicht te krijgen in het HOE.

2.3 Context diagram



Figuur 2 Context diagram

2.3.1 Elementen in Context diagram

SDW-script

Een script dat binnen SDW op basis van de *data-dictionary* het *GW96 stuurbestand* genereert.

editor

Een editor waarmee bilaterale afspraken kunnen worden vastgelegd in het *bilaterale stuurbestand.*

uitwisselingsformaat generator

Een programma dat op basis van het *GW96 stuurbestand* en het *bilaterale stuurbestand* het *uitwisselingsformaat* genereert.

applicatie 1 en 2

Een applicatie bestaande uit een gebruikersdeel en de stekkerdoos waarmee op basis van *uitwisselingsformaat* gegevens worden uitgewisseld (=lezen en schrijven naar *uitwisselingsbestanden*). Applicaties kunnen lezen en/of schrijven in een of meerdere uitwisselingsbestanden.

2.3.2 Gegevensstromen in Context diagram

applicatie 1 en 2 invoer

Dit zijn alle invoergegevens die noodzakelijk zijn voor het uitvoeren van applicatie 1 en 2.

edit opdrachten

Dit zijn alle edit-opdrachten die de gebruiker uitvoert voor het samenstellen van het *bilaterale stuurbestand.*

gebruikers opdracht

Dit zijn alle gebruikersopdrachten voor het uitvoeren van de diverse processen.

2.3.3 Buffers in Context diagram

SDW data-dictionary

Deze buffer bevat alle gegevens (entiteiten, attributen, relaties, etc.) die betrekking hebben op het Adventus-datamodel.

GW96 stuurbestand

Deze buffer bevat de beschrijving van de tabellen (tabellen en entiteiten) van het Adventus datamodel.

bilaterale stuurbestand

Deze buffer bevat de beschrijving van de tabellen (tabellen en entiteiten) afkomstig uit de bilaterale afspraken.

uitwisselingsformaat

Deze buffer bevat alle entiteiten (en attributen) van het *GW96 stuurbestand* en het *bilaterale stuurbestand*, waarbij de entiteiten uit het *bilaterale stuurbestand* de voorrang genieten in het geval het gaat om gelijkluidende entiteiten.

uitwisselingsbestand

Deze buffer bevat de gegevens welke uitgewisseld worden.

logfile

Deze buffer bevat de foutmeldingen die door de stekkerdoos worden gegenereerd. Foutmeldingen gegeneerd door Nefis worden niet in de logfile geschreven, maar direct doorgegeven aan de applicatie.

2.3.4 Gebeurtenissenlijst

1. M.b.v. een script kan uit de SDW data-dictionary het *GW96 stuurbestand* worden gegeneerd. Dit bestand bevat een beschrijving van alle binnen Adventus gedefinieerde entiteiten en attributen.

2. M.b.v. een editor kan het *bilaterale stuurbestand* worden gemaakt, waarin aanpassingen op het Adventus datamodel kunnen worden aangegeven, alsmede 'eigen' entiteiten en waardereeksen worden gedefinieerd.

3. M.b.v. de uitwisselingsformaat generator worden de twee stuurbestanden samengevoegd tot één bestand (het *uitwisselingsformaat*), waarin alle entiteiten, waardereeksen en attributen zijn beschreven. Dit *uitwisselingsformaat* wordt gebruikt door Stekkerdoos Water voor het schrijven naar een *uitwisselingsbestand*.
   nb Het uitwisselen van gegevens gebeurt middels het *uitwisselingsbestand* (=data) en het *uitwisselingsformaat* (=beschrijving).
   nb In het geval gegevens alleen gelezen behoeven te worden, kan bij het uitwisselen worden volstaan met het *uitwisselingsbestand* omdat alle gebruikte entiteiten, waardereeksen en attributen beschreven zijn op het *uitwisselingsbestand*.

4. M.b.v. de stekkerdoos en het *uitwisselingsformaat* kunnen binnen applicatie 1 en 2 worden gelezen en geschreven naar meerdere *uitwisselingsbestanden*.

2.3.5 De afbakening

Op basis van het context diagram wordt vastgesteld dat de volgende onderdelen deel uit maken van Stekkerdoos Water:

1. de stekkerdoos (=software bibliotheek) ,
2. uitwisselingsformaat generator en
3. de buffers: *GW96 stuurbestand*, *bilaterale stuurbestand*, *uitwisselingsformaat*, *logfile* en *uitwisselingsbestand*.

nb De structuur en het format van de buffers *logfile, uitwisselingsformaat* en *uitwisselingsbestand* zullen in het Technisch Ontwerp worden vastgelegd.

nb De structuur en het format van de buffers *GW96 stuurbestand* en *bilaterale stuurbestand* is in eerder ontwerp vastgelegd en zal niet wijzigen.

3 Het gedrag van het systeem

3.1 Inleiding

De tweede stap is het (globaal) vaststellen van de gewenste functionaliteit; de eisen WAT het systeem moet kunnen, dient te worden vastgelegd.
Om daar inzicht in te krijgen is het gedragsmodel (behaviour model) opgesteld. Binnen het gedragsmodel komen aan de orde:

1. de functies met de gegevensstromen en gegevensopslag waarmee het systeem op gebeurtenissen (events) uit de omgeving reageert (=Dataflow Diagrammen) en
2. de (globale) werking van de functies (=beschrijving).

ad 1 en 2.
Bij de functiebeschrijvingen komen de volgende items aan bod:

BESCHRIJVING	= wat doet de functie
TRIGGER	= wanneer wordt de functie uitgevoerd
INPUT	= waar komen de gegevens vandaan
OUTPUT	= waar gaan de gegevens naar toe

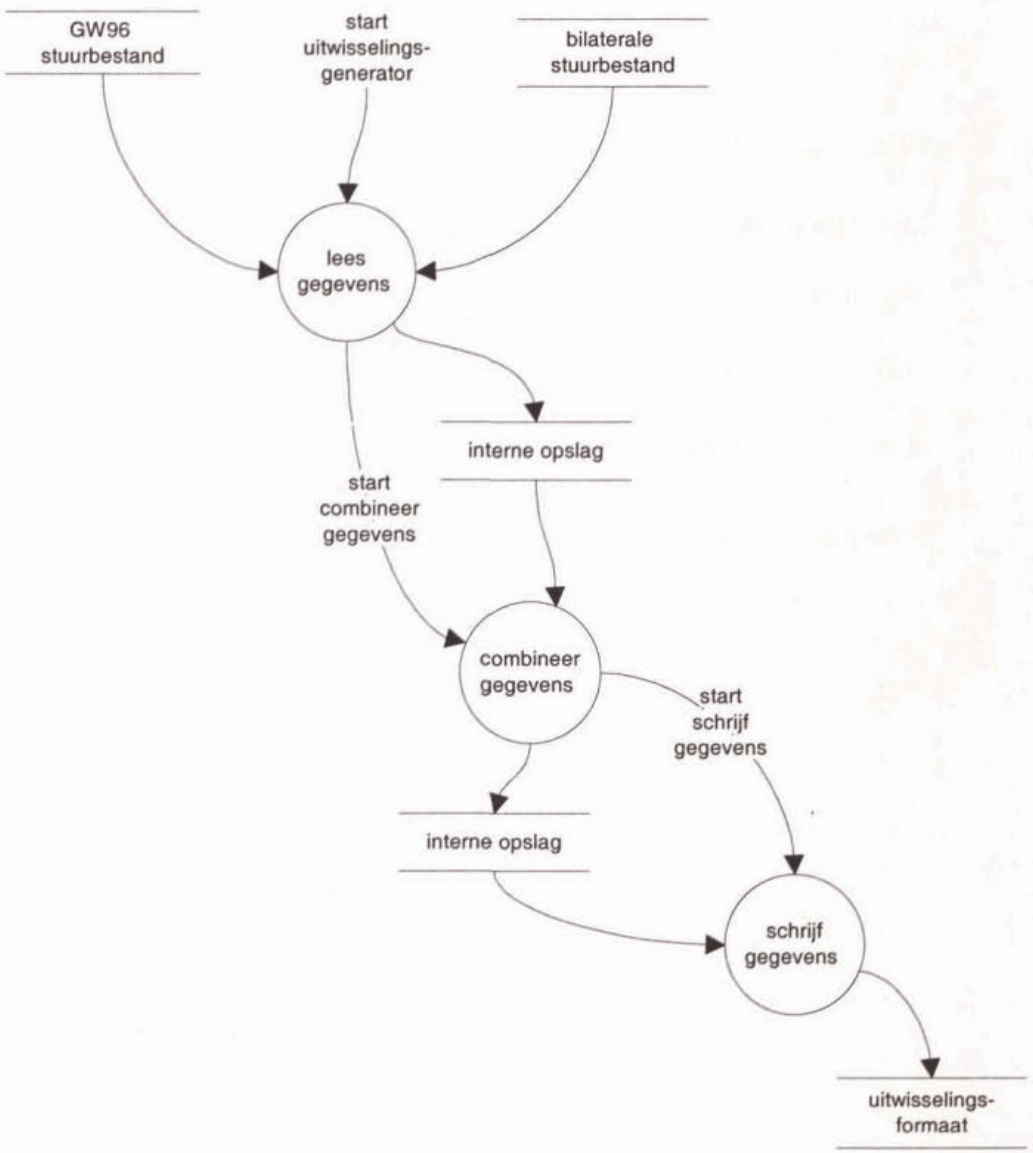
Formeel moeten in het gedragsmodel ook de volgende twee punten aan de orde komen:

1. de gegevens en hun onderlingen relatie (=Entity Relation Diagram) en
2. de samenstelling en betekenis van de gegevens (=Data Dictionary).

Omdat de opslag van gegevens en hun onderlinge relatie de basis vormen voor het goed functioneren (lees: goed performen) zal in dit Functioneel Ontwerp daar slechts zeer globaal aandacht aan worden besteedt. Deze punten zullen in detail worden uitgewerkt in het Technisch Ontwerp.

3.2 Uitwisselingsformaat generator

3.2.1 Dataflow diagram



Figuur 3 Uitwisselingsformaat generator

3.2.2 Functies

Lees gegevens

BESCHRIJVING	Met deze functie worden de entiteit-, attribuut- en waardenreeks beschrijvingen gelezen en opgeslagen in geheugen.
TRIGGER	start uitwisselingsgenerator
INPUT	GW96 stuurbestand en bilaterale stuurbestand
OUTPUT	interne opslag

Combineer gegevens

BESCHRIJVING	Met deze functie worden de opgeslagen gegevens gecontroleerd en gecombineerd waarna deze ook weer opgeslagen worden in geheugen.
TRIGGER	start combineer gegevens
INPUT	interne opslag
OUTPUT	interne opslag

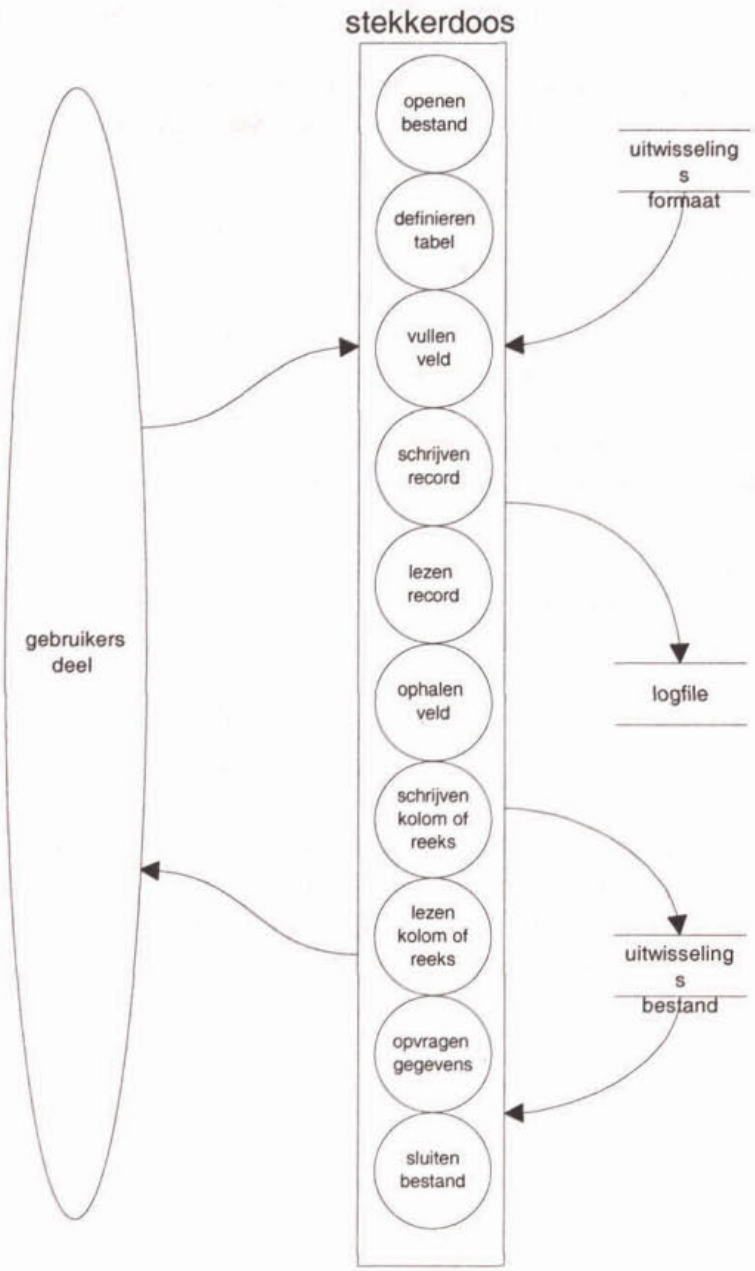
Schrijf gegevens

BESCHRIJVING	Met deze functie worden de opgeslagen gegevens weggeschreven naar het bestand: uitwisselingsformaat.
TRIGGER	start schrijf gegevens
INPUT	interne opslag
OUTPUT	uitwisselingsformaat

3.3 De stekkerdoos

3.3.1 Context diagram

Inzoomen (in het context diagram) op applicatie 1 levert het volgende beeld.
Hierin is te zien dat het gebruikersdeel van applicatie 1 communiceert met het uitwisselingsbestand via de stekkerdoos-functies waarbij de Stekkerdoos alle lees- en schrijfopdrachten verzorgt.

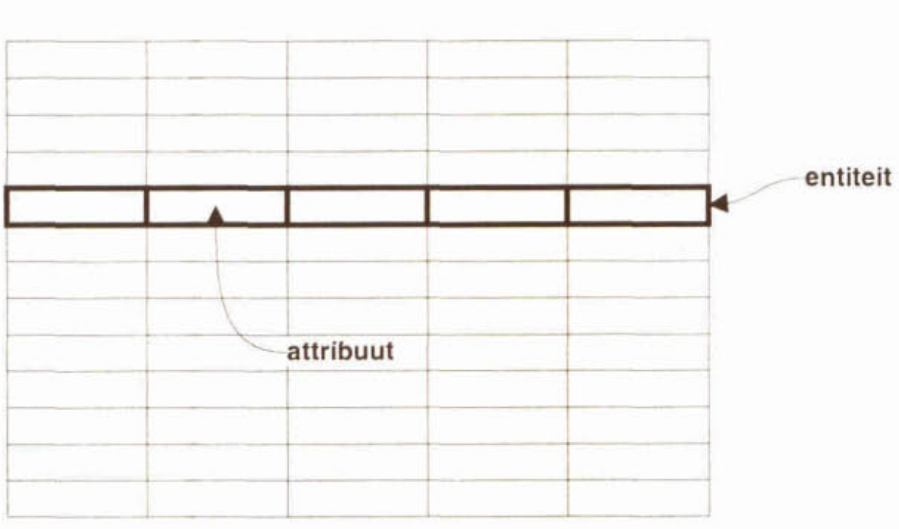


Figuur 4 Context diagram (stekkerdoos)

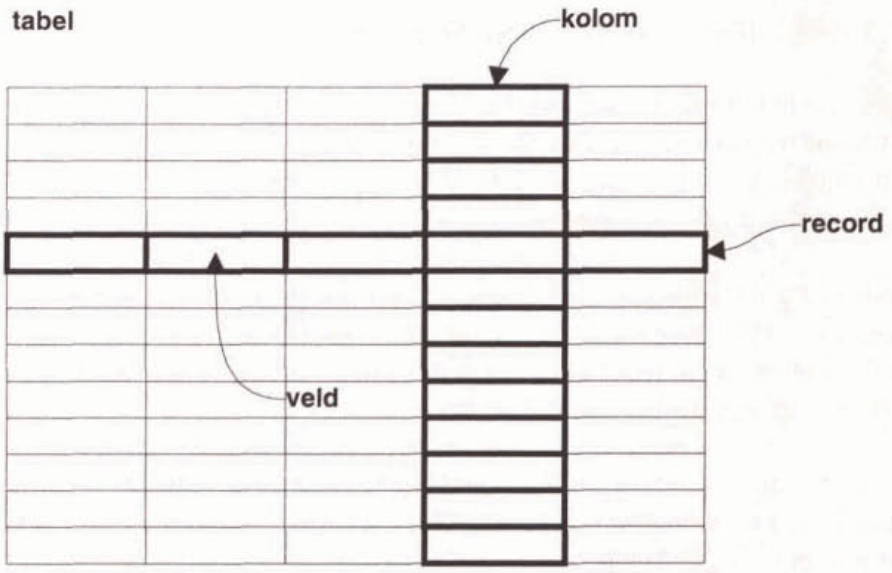
Bij het context diagram op de vorige pagina kan nog worden opgemerkt dat het schrijven van een kolom zeer veel overeenkomst vertoond met schrijven van attributen van een waardenreeks. Daarom zijn beide in dit ontwerp binnen een functie geplaatst. Hetzelfde geldt voor lezen kolom of reeks.
In de realisatie zal voor de duidelijkheid ten behoeve van de gebruiker/programmeur gekozen worden voor twee verschillende functienamen.
Ook de functie 'definiëren tabel' is vrijwel identiek voor waardenreeks en tabel en hiervoor gelden dezelfde opmerkingen als hierboven gemaakt.
Het begrip 'waardenreeks' wordt verder uitgelegd onder paragraaf 3.3.3.

3.3.2 Begrippen

Binnen de datamodellering wordt gebruik gemaakt van de begrippen: entiteittype, entiteit en attribuut.
Een entiteittype is een collectie, of een verzameling, van entiteiten.
Een entiteit wordt beschreven door een aantal attributen (kenmerken).



Voor de fysieke opslag worden deze begrippen vertaald naar tabellen, waarbij een tabel opgebouwd is uit meerdere kolommen, records en velden.
Een record (=entiteit) is één voorkomen binnen de gehele tabel (=entiteittype).
Een record (=entiteit) zelf is opgebouwd uit meerdere velden (=attributen).



In dit ontwerp zal niet meer gesproken worden in termen van entiteittype entiteit en attribuut, maar in termen van *tabel*, *record*, *veld* en *kolom* als het betrekking heeft op het Adventus gegevensmodel.

Bij het ontwerp van de eerste versie van Stekkerdoos Water is onderkend dat ook reeksen waarden, zoals tijdreeksen, uitgewisseld moeten kunnen worden. Destijds is de term *waarden-reeks* ingevoerd voor een bij elkaar behorende verzameling waarden, opgeslagen in arrays. Een andere tijdreeks is een nieuwe verzameling waarden, van het type tijdreeks en met een unieke naam.
Daarmee is een Stekkerdoos waardenreeks iets anders dan het entiteittype 'waardereeks' uit het Adventus gegevensmodel! In Adventus worden bij alle (meet)waarden opgeslagen in de tabel 'Meetwaarde' en is er een relatie naar een tijdreeks.

De Stekkerdoos waardenreeks wordt in de volgende paragraaf 3.3.3. uitvoerig uitgelegd. Ook daar komen we tot begrippen entiteittype en attributen. Daarmee zouden we ook voor waardenreeksen kunnen spreken van tabellen en velden.
Echter het enigszins bijzondere karakter van waardenreeks als bij elkaar behorende arrays van waarden en ook de bekendheid uit vorige versies van de Stekkerdoos hebben er toe geleid om voor waardenreeksen als entiteittype de termen waardenreeks en attribuut te gebruiken in plaats van tabel en veld.
Waardenreeksen worden onder een unieke naam op het uitwisselingsbestand opgeslagen.

3.3.3 Waardenreeksen in de Stekkerdoos

Uitleg van het begrip waardenreeks
In het functioneel ontwerp van Stekkerdoos versie 1.0 is het begrip *waardenreeks* reeds geïntroduceerd. De daar gegeven definitie zegt: "waardenreeksen zijn (lange) reeksen van getallen met een vaste structuur."

In dit document willen we het begrip *waardenreeks* opnieuw beschrijven, een verband leggen met entiteittypen en van daaruit aangeven hoe waardenreeksen op het bilaterale stuurbestand worden gedefinieerd. Daarbij wordt aangegeven hoe de programmeur of gebruiker met waardenreeksen kan omgaan.

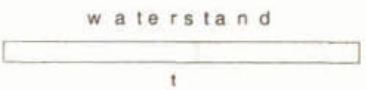
Als eerste kunnen we zeggen dat waardenreeksen attributen bezitten (die de kenmerken weergeven). De attributen van de waardenreeks hebben een zekere afmeting, terwijl de waardenreeks zelf een lengte heeft.
Voorbeelden waardenreeksen zijn:

- Tijdreeks: heeft een lengte en bestaat uit de attributen Datum, Tijd, Waarde;
- Polygoon: heeft een lengte en bestaat uit de attributen X-coördinaat, Y-coördinaat;
- Grid: heeft een 'lengte' bestaat uit het attribuut Coördinaten, met de afmeting m*n;
- Rooster: heeft een 'lengte' en bestaat uit de attributen X-coördinaat enY-coördinaat, ieder met de afmeting n*m;
- 3Dblok: heeft een 'lengte' en het attribuut Parameters met de afmeting m*n*t

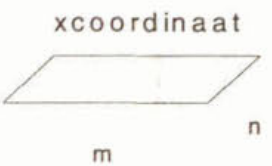
Wel is het zo dat ieder attribuut van een waardenreeks exact dezelfde afmeting heeft. Aan het einde van deze paragraaf wordt verder ingegaan op de begrippen lengte voor waardenreeks en afmeting voor attribuut.

Een aantal voorbeelden van waardenreeksen (met één attribuut) zijn:

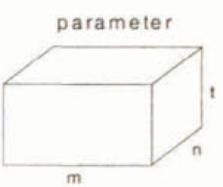
- een tijdreeks met waterstanden voor een bepaalde locatie



- een matrix met de x-coördinaten van een 2-dimensionaal rekenrooster,



- een 3d matrix met de parameterwaarden (bijv. waterstanden) van een2-dimensionaal rekenrooster in de tijd,



Meer algemeen kan worden gezegd: een n-dimensionale matrix met waarden, waarbij n loopt van 1..5. De applicatie en niet de stekkerdoos kent de betekenis van de structuur!

Formele beschrijving van waardenreeks als entiteittype met attributen
Kijken we nu op een formeel of logisch niveau naar waardenreeksen dan kun je ook zeggen dat de waardenreeksen entiteittypen zijn met attributen die de kenmerken van de waardenreeks weergeven. Er zijn dan entiteittypen als bijvoorbeeld:

- Tijdreeks,
- Polygoon,
- Rooster,
- Grid

Allen hebben als attributen matrices of anders gezegd: verzamelingen met waarden.
Dus hier is een attribuut meer dan één afzonderlijke waarde (zoals bij 'gewone' entiteiten). Op deze wijze is er in de beschrijving op het stuurbestaand geen onderscheid tussen 'gewone' entiteiten en waardenreeksen.

Bij 'gewone' entiteittypen ontstaan nieuwe voorkomens via een sleutelwaarde en een nieuw record in de tabel. De Stekkerdoos Water zorgt voor opslag in het uitwisselingsbestand via functies als *creëer tabel* van een entiteittype, *vullen velden*, *vullen sleutelvelden* en *schrijf record*. Natuurlijk zijn er ook de leesfuncties.
Bij de waardenreeksen ontstaan nieuwe voorkomens via een unieke naam (=sleutel) en het aangeven van het entiteittype waarvan deze is afgeleid. De Stekkerdoos Water zorgt voor opslag in het uitwisselingsbestand via functies als *creëer waardenreeks* en *schrijf (waarden)reeks*.

De waardenreeksen hebben als attributen *verzamelingen (matrices)* met waarden en niet *één afzonderlijke* waarde, zoals bij de gewone entiteittypen. Deze attributen zijn allemaal wel van éénzelfde type (1d, 2d, 3d... n-dimensionale matrix), zoals opgegeven bij de beschrijving van de waardenreeks.
De attributen (=velden) kunnen nu niet meer gevuld worden met een functie als *vullen veld* of *ophalen veld* en vervolgens *lees record* of *schrijf record*. Omdat de attributen van waardenreeksen veelal een forse hoeveelheid waarden zijn, wordt een dergelijk attribuut in één keer naar file geschreven of er van gelezen. Deze functies zouden een naam als *schrijf waardenreeks-attribuut* kunnen hebben en idem voor het lezen. Om pragmatische reden is gekozen voor de naam *schrijf reeks* of *lees reeks* omdat attributen al snel reeksen worden genoemd (denk aan het veel gebruikte voorbeeld Tijdreeks)

Het schrijven en lezen van waardenreeksen
Hierbij kunnen de volgende stappen worden onderscheiden:

1. Als eerste is er de definitie van de waardenreeks als entiteittype met een naam en met attributen die een naam en afmeting hebben. Deze worden opgegeven in het zogenaamde bilaterale stuurbestand.
2. Creëer daarna via een functie als bijvoorbeeld *crewrd* een unieke waardenreeks met naam en van een bepaald entiteittype; bijvoorbeeld T1 van type Tijdreeks.
3. Schrijf dan de waarden (= alle attributen) van T1 naar het uitwisselings-bestand, dat wil zeggen schrijf de afzonderlijke attributen Datum, Tijd, Waarde waaruit de tijdreeks bestaat.
4. Voor andere waardenreeksen, als bijvoorbeeld van het type Rooster en 3Dblok, geldt dezelfde procedure.

Relatie met Adventus

Adventus kent ook het entiteittype '*waardereeks*', zie ERD metingen.2 Deze entiteit heeft attributen voor het vastleggen van gegevens omtrent een waardenreeks. Het entiteittype kan bijvoorbeeld worden uitgebreid met attributen voor de unieke naam (sleutel) van een waardenreeks en zijn type, zoals hierboven beschreven. Daarmee wordt een relatie gelegd naar de specifieke waardenreeks zoals we die kennen in de Stekkerdoos.

Voor alle duidelijkheid: hierboven is het begrip waardenreeks beschreven zoals gebruikt binnen de Stekkerdoos Water. Er ontstaan entiteittypen als Tijdreeks, Polygoon, Rooster, Grid met als attributen verzameling matrices met waarden. Het betreft hier steeds de opslag van de waarden. Het entiteittype '*waardereeks*' uit Adventus geeft aanvullende informatie over waardenreeksen en een relatie naar de daadwerkelijke verzameling waarden.

Relaties tussen entiteiten en waardenreeksen

Een waardenreeks wordt geïdentificeerd door zijn unieke naam. Hiervoor zijn tabellen aanwezig binnen de Stekkerdoos Water. Een waardenreeks behoeft niet gekoppeld te zijn aan een andere entiteit. Anderzijds is het mogelijk dit wel te doen via bijvoorbeeld het Adventus entiteittype '*waardereeks*' (zie hierboven).
Het is ook mogelijk relaties te leggen tussen waardenreeksen en andere entiteit-typen. Daarvoor moeten deze entiteittypen worden uitgebreid met nieuwe attributen o.a. voor de naam van de waardenreeks.

Afmeting/lengte van waardenreeksen

Hierboven is reeds opgemerkt dat de attributen van waardenreeksen een vaste structuur hebben. Aan de andere kant is het gewenst (prettig) dat bij het creëren van een waardenreeks toch een afmeting (een lengte) kan worden opgegeven. Via het uitwerken van voorbeelden willen we het verband tussen *attribuut-dimensie* (=afmeting van het attribuut) en de *lengte van een waardenreeks* aangeven.

In het eerder genoemde voorbeeld Tijdreeks hebben we te maken met een 1-dimensionale reeksen als attributen. Dit is ook te zien als een 1-dimensionale array.
We komen tot deze 1-dimensionale array door bij de definitie van entiteittype Tijdreeks bij de attributen de attribuut-dimensie 1 op te geven (= 1 enkel element van de 1d array). Bij het creëren van een tijdreeks met bijv. naam T1 geven we de lengte van de waardenreeks op, bijvoorbeeld 102. De combinatie van attribuut-dimensie 1 met lengte 102 geeft de 1d array (van 102) voor de attributen Datum, Tijd en Waarde van Tijdreeks T1.

In het geval een entiteittype Punt met als attribuut "X-Y" voor de x,y coördinaten bevat de definitie van Punt het attribuut "X-Y" met attribuut-dimensie 2. Hierin worden de x en y waarden opgeslagen. Bij de creatie van een waardenreeks met naam Punt-A krijgt deze de waardenreeks-lengte van bijvoorbeeld 145, om 145 punten weg te kunnen schrijven. Bij het schrijven naar het uitwisselingsbestand hebben we dus te maken met een 2d array van 2 * 145.

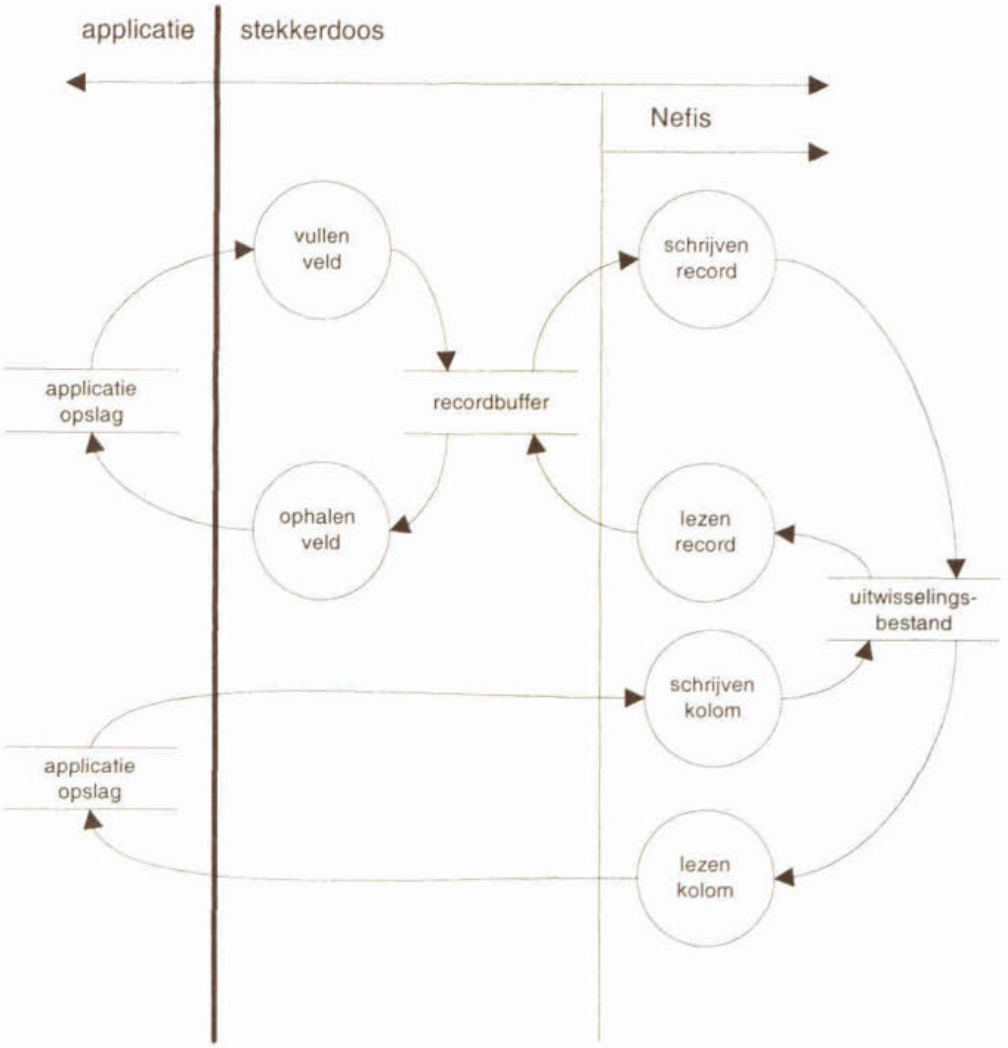
Voor de al eerder genoemde voorbeelden als Grid en Rooster geldt eenzelfde aanpak. Bij de definitie wordt als attribuutdimensie opgegeven bijvoorbeeld 100 * 275 voor het grid of rooster. Bij de creatie wordt dan een waardenreekslengte van 1 opgegeven. Daarmee krijgt de unieke waardenreeks met naam GRID-X van entiteittype GRID voor het attribuut "Coördinaten" een 3-dimensionale array van 100 * 275 * 1 , die naar het uitwisselingsbestand geschreven wordt.
Zo kunnen met deze getalvoorbeelden voor waardenreeks R1 van entiteittype Rooster de twee attributen "X-coördinaat" en "Y-coördinaat" als twee afzonderlijke 3-d arrays van 100 * 275 * 1 worden weggeschreven.
Voor het genoemde voorbeeld 3Dblok geven we bij de definitie een vaste attribuut-dimensie op (dat is het rekenrooster waarop de parameterwaarden zijn gebaseerd), zeg 112 * 245. Bij de creatie van een blok met parameterwaarden geven we de waardenreekslengte op van bijvoorbeeld 7. Hiermee worden dan in een 3-d matrix van 112 * 245 * 7 , de 7 gewenste parameters (gebaseerd op het grid van 112 * 245) naar het uitwisselingsbestand geschreven.

NB
Bij het wegschrijven en teruglezen van arrays naar/uit het uitwisselingsbestand wordt de opslag van de waarden in de arrays beschouwd als een aaneengesloten geheugenblok. De kennis over de volgorde van de afzonderlijke waarden is aanwezig in de applicatieprogrammatuur.

3.3.4 Het idee

3.3.4.1 records

Om de performance van de huidige stekkerdoos te verbeteren zal het lezen en schrijven naar het uitwisselingsbestand beperkt moeten worden. Dit kan bereikt worden door het uitwisselingsbestand niet per veld (=attribuut) maar per record (=entiteit) te benaderen.
Om dit mogelijk te maken zal een recordbuffer (=memory) bijgehouden moeten worden waarin het bewerken van de velden van een record plaatsvindt. Het recordbuffer wordt in zijn geheel gelezen of geschreven van of naar het uitwisselingsbestand. Hiervoor worden de Nefis-functies gebruikt.



Figuur 5 Recordbuffer

Deze werkwijze biedt een aantal voordelen:

- het lezen en schrijven van afzonderlijke velden van een record gebeurt 'in memory' en is dus snel

- voor het lezen en schrijven van een record kan 'onder water' een splitsing worden gemaakt in verschillende type array's (characters, integers, reals)
- de applicatieprogrammeur hoeft zich nu niet te bekommeren om het type (character, integer of real)
- de lees- en schrijfroutines voor Nefis worden op deze manier op een zo laag mogelijk niveau geïmplementeerd. Vervanging door andere lees- en schrijfroutines ten behoeve van een ander filetype (bijvoorbeeld NetCDF, HDF, ASCII) wordt daardoor vereenvoudigd.

3.3.4.2 kolommen

Om het gebruik van een tabel maximaal toegankelijk te maken moet het lezen en schrijven niet beperkt blijven tot afzonderlijke records; het zal ook mogelijk moeten zijn afzonderlijke kolommen te kunnen lezen en schrijven **!!!**

Het lezen en schrijven van kolommen biedt niet alleen de applicatieprogrammeur maximale vrijheid, maar effent tevens de weg voor maximale performance omdat een gehele kolom (= één array) met één opdracht gelezen of geschreven kan worden.

Voor een 'normale' tabel zullen in de meeste gevallen steeds per record worden gelezen of geschreven; waarbij de mogelijkheid bestaat om controles uit te voeren op uniciteit van sleutelvelden. Het schrijven van een tabel met 1000 records met 20 velden bestaat uit 1000 schrijfopdrachten.

Er is echter ook de mogelijkheid om de gegevens per kolom te schrijven. In dat geval zal het schrijven van dezelfde tabel bestaan uit 20 schrijfopdrachten. Voordeel is minder schrijfopdrachten, is performanceverbetering. Nadeel is dat de gegevens eerst verzameld (=opgeslagen) moeten worden en dat er geen controle op uniciteit van sleutelvelden kan plaatsvinden.

3.3.5 Functionaliteit van de stekkerdoos

Een en ander levert de volgende gewenste functionaliteit op (zie ook Context Diagram, hoofdstuk 3.3.1).

openen bestand

Met deze functie kunnen 1 of meerder uitwisselingsbestanden worden geopend met als doel: lezen en/of schrijven.
Voor het schrijven van meerdere uitwisselingsbestanden kan gebruik gemaakt worden van één en hetzelfde uitwisselingsformaat, maar ook van verschillende uitwisselingsformaten.

definiëren tabel

Met deze functie wordt een tabel gedefinieerd. Het definiëren gebeurt middels het opgeven van een tabelnaam. De benodigde veldspecificaties worden uit het uitwisselingsformaat gelezen.

nb Omdat er geen verschil is tussen een 'normale' tabel en een waardenreekstabel worden beide tabellen met deze functie gedefinieerd.

vullen veld

Met deze functie wordt één veld van een record worden gevuld (=in het recordbuffer geplaatst). Dit gebeurt door middel van veldnaam en veldwaarde. Ook sleutelvelden worden op deze manier gevuld.

schrijven record

Met deze functie wordt één record naar een tabel geschreven, m.a.w. het recordbuffer (=alle velden) wordt in zijn geheel naar de tabel te geschreven. Dit gebeurt door middel van tabelnaam en de methode van schrijven (update, toevoegen, controle sleutelveldwaarden).

lezen record

Met deze functie wordt één record van een tabel gelezen, m.a.w. het recordbuffer wordt gevuld met alle velden uit een record. Dit gebeurt door middel van tabelnaam, sleutelveldnamen, sleutelveldwaarden en de methode van lezen (eerste of laatste record of zoeken via sleutel).

ophalen veld

Met deze functie wordt één veld van een record gelezen (=uit het recordbuffer gelezen). Dit gebeurt door middel van veldnaam en veldwaarde.

schrijven kolom of reeks

Met deze functie wordt één kolom rechtstreeks (zonder tussenkomst van het recordbuffer) naar een tabel geschreven. Dit gebeurt door middel van tabelnaam en kolomnaam (=veldnaam).

nb Er is geen functioneel verschil tussen het schrijven van een kolom (in een 'normale' tabel) of het schrijven van een veld in een waardenreekstabel. In het ene geval wordt een 1-dimensionale dataset geschreven in meerdere velden (in dezelfde kolom) en in het andere geval wordt een meer-dimensionale dataset geschreven in 1 veld.

lezen kolom of reeks

Met deze functie wordt één kolom van een tabel gelezen. Dit gebeurt door middel van tabelnaam en kolomnaam (=veldnaam).

nb Analoog aan het schrijven van een kolom of reeks is en geen functioneel verschil tussen het lezen van een kolom of reeks.

opvragen gegevens

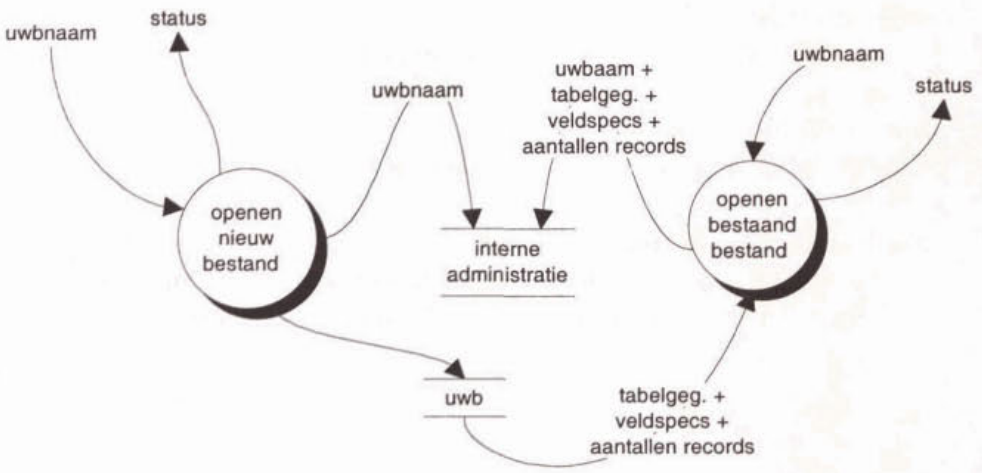
Met deze functie worden gegevens opgevraagd over de inhoud van het uitwisselingsbestand.

sluiten bestand

Met deze functie worden alle uitwisselingsbestand gesloten.

nb *In alle hierna volgende dataflow diagrammen is de buffer 'logfile' i.v.m. overzichtelijkheid weggelaten. Bij de detaillering in het Technisch Ontwerp zal deze wel worden meegenomen.*

3.3.6 Openen bestand



Figuur 6 Openen bestand

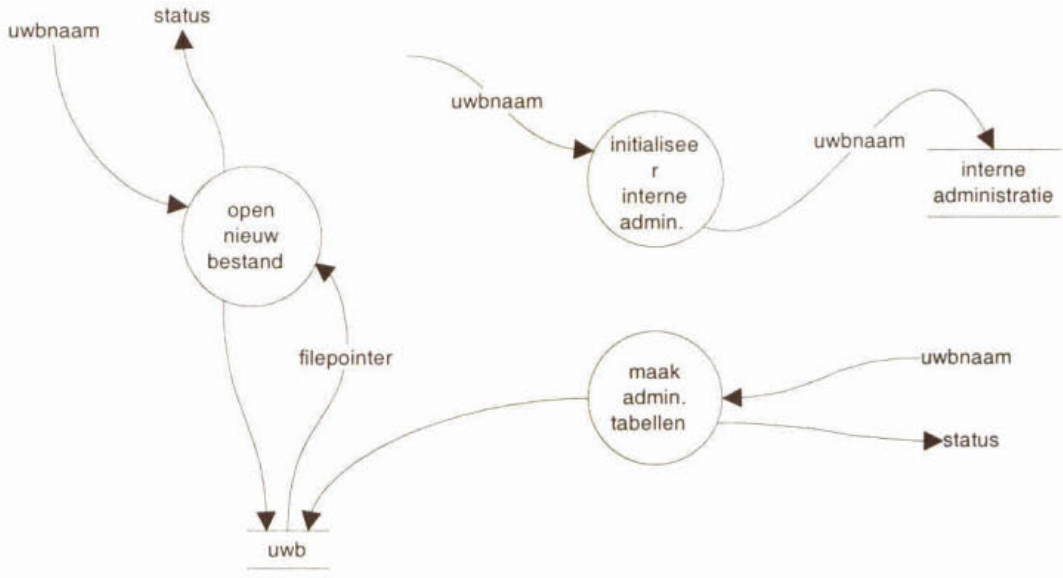
openen nieuw bestand

Met deze functie wordt een nieuw uitwisselingsbestand geopend en de interne administratie geïnitialiseerd.

openen bestaand bestand

Met deze functie wordt een bestaand uitwisselingsbestand geopend en de interne administratie geïnitialiseerd op basis van het uitwisselingsbestand.

3.3.6.1 openen nieuw bestand



Figuur 7 Openen nieuw bestand

open nieuw bestand

Met deze functie wordt een nieuw uitwisselingsbestand (uwbnaam) geopend.

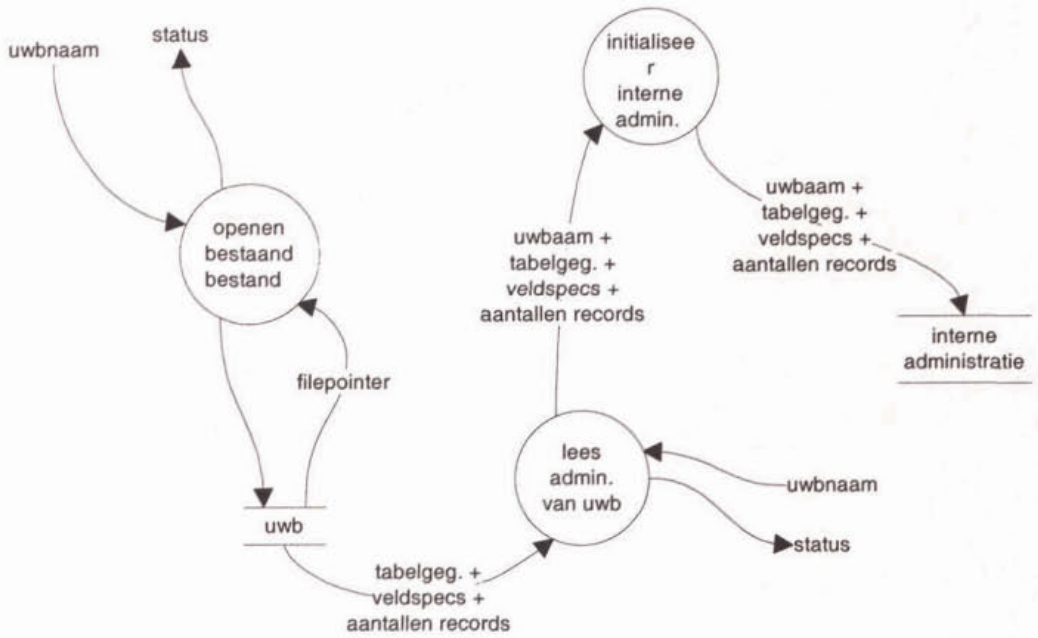
initialiseer interne admin.

Met deze functie wordt de *interne administratie* geïnitialiseerd.

maak administratie tabellen

Met deze functie wordt de tabellen op het uitwisselingsbestand (*uwb*) gedefinieerd, zonder dat deze worden gevuld; dit gebeurt bij 'maken tabel'.

3.3.6.2 openen bestaand bestand



Figuur 8 Openen bestaand bestand

openen bestaand bestand

Met deze functie wordt een bestaand uitwisselingsbestand (*uwb*) geopend.

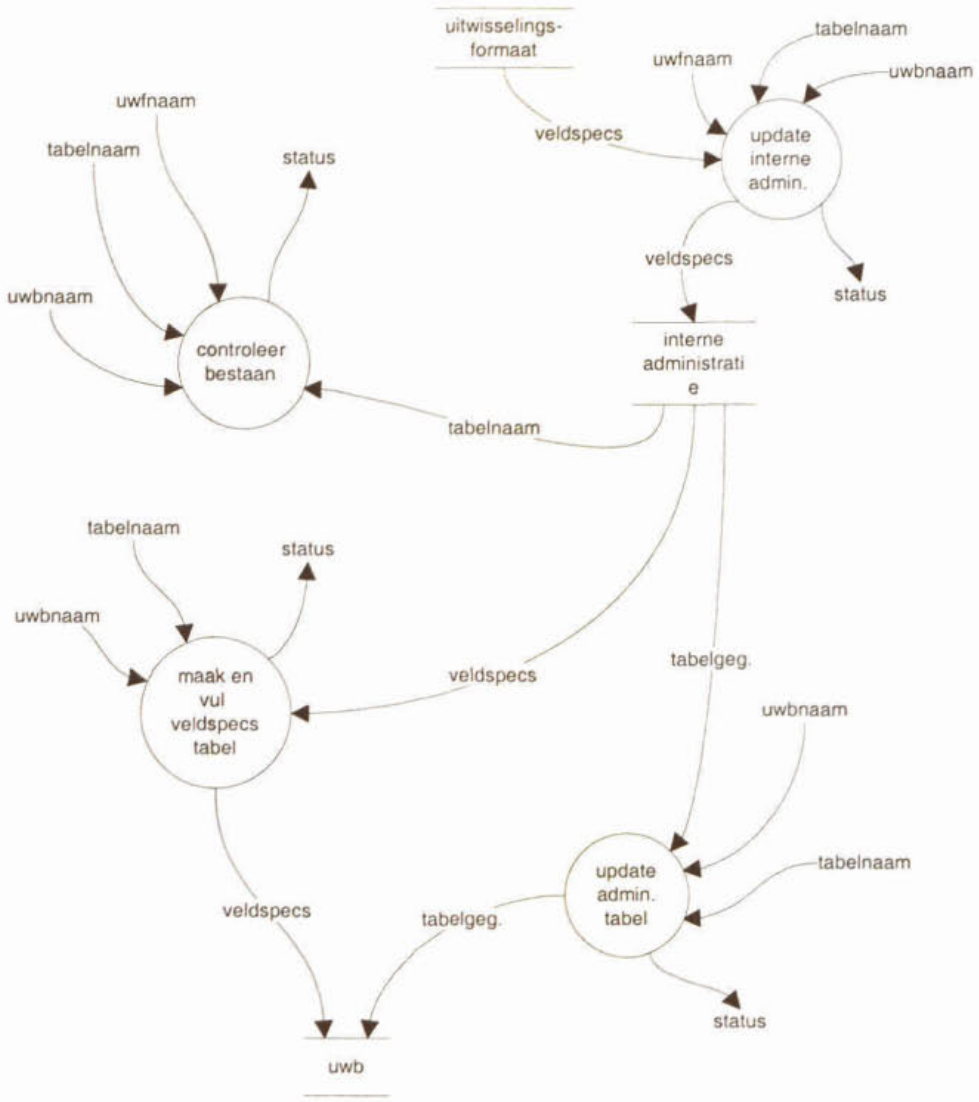
lees admin. van uwb

Met deze functie worden de administratietabellen (=beschreven entiteiten en attributen) van het *uwb* gelezen.

initialiseer interne admin.

Met deze functie wordt de *interne administratie* geïnitialiseerd.

3.3.7 Definiëren tabel



Figuur 9 Definiëren tabel

controleer bestaan

Met deze functie wordt gecontroleerd of de tabelnaam al bestaat op het *uwb*. Hiervoor wordt de tabelnaam vergeleken met alle tabelnamen in de *interne administratie*. Vervolgens wordt gecontroleerd of de tabelnaam voorkomt in het *uitwisselingsformaat*.

update interne admin.

Met deze functie wordt de interne administratie bijgewerkt; d.w.z. de tabelnaam en de veldspecificaties worden toegevoegd.

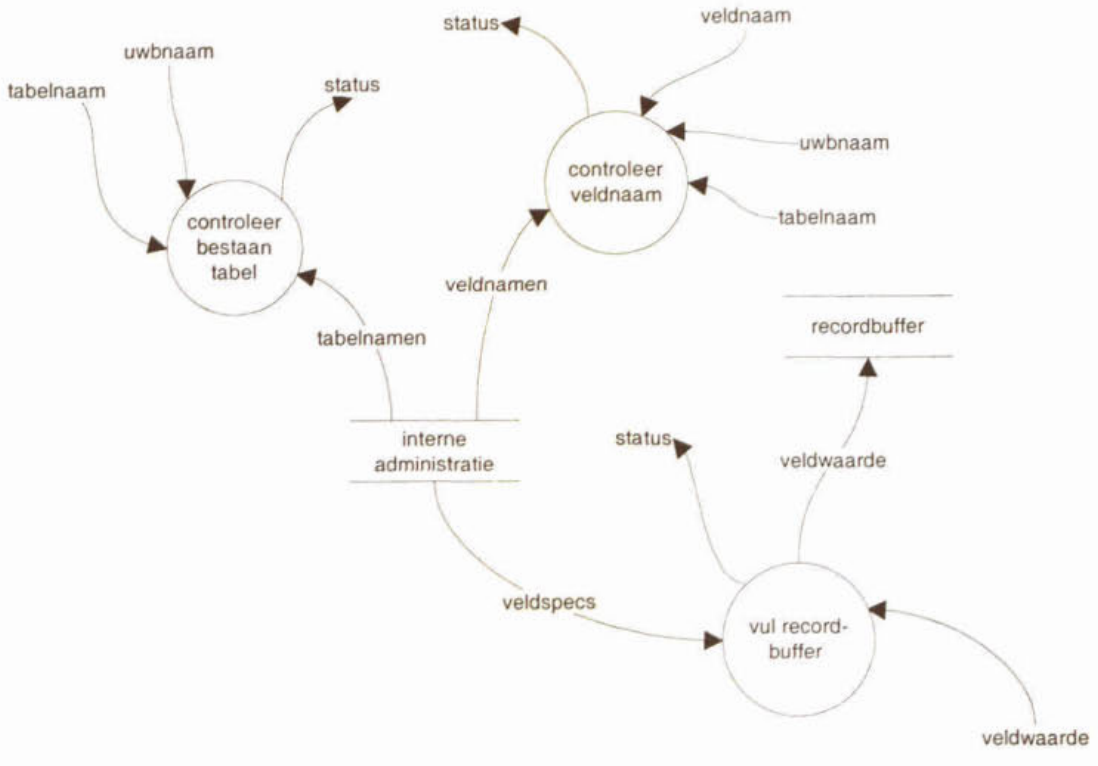
maak en vul veldspecs tabel

Met deze functie wordt de tabel gedefinieerd waarin de veldspecificaties van de betreffende tabel opgeslagen kunnen worden. Nadat de tabel is gedefinieerd worden de veldspecificaties daarin opgeslagen.

update tabel admin.

Met deze functie wordt de administratie tabel bijgewerkt; d.w.z. dat de tabelnaam wordt daarin weggeschreven.

3.3.8 Vullen veld



Figuur 10 Vullen veld

controleer bestaan tabel

Met deze functie wordt gecontroleerd of de tabelnaam bestaat op het *uwb*. Hiervoor wordt de tabelnaam vergeleken met alle tabelnamen in de *interne administratie* voor het betreffende uwb.

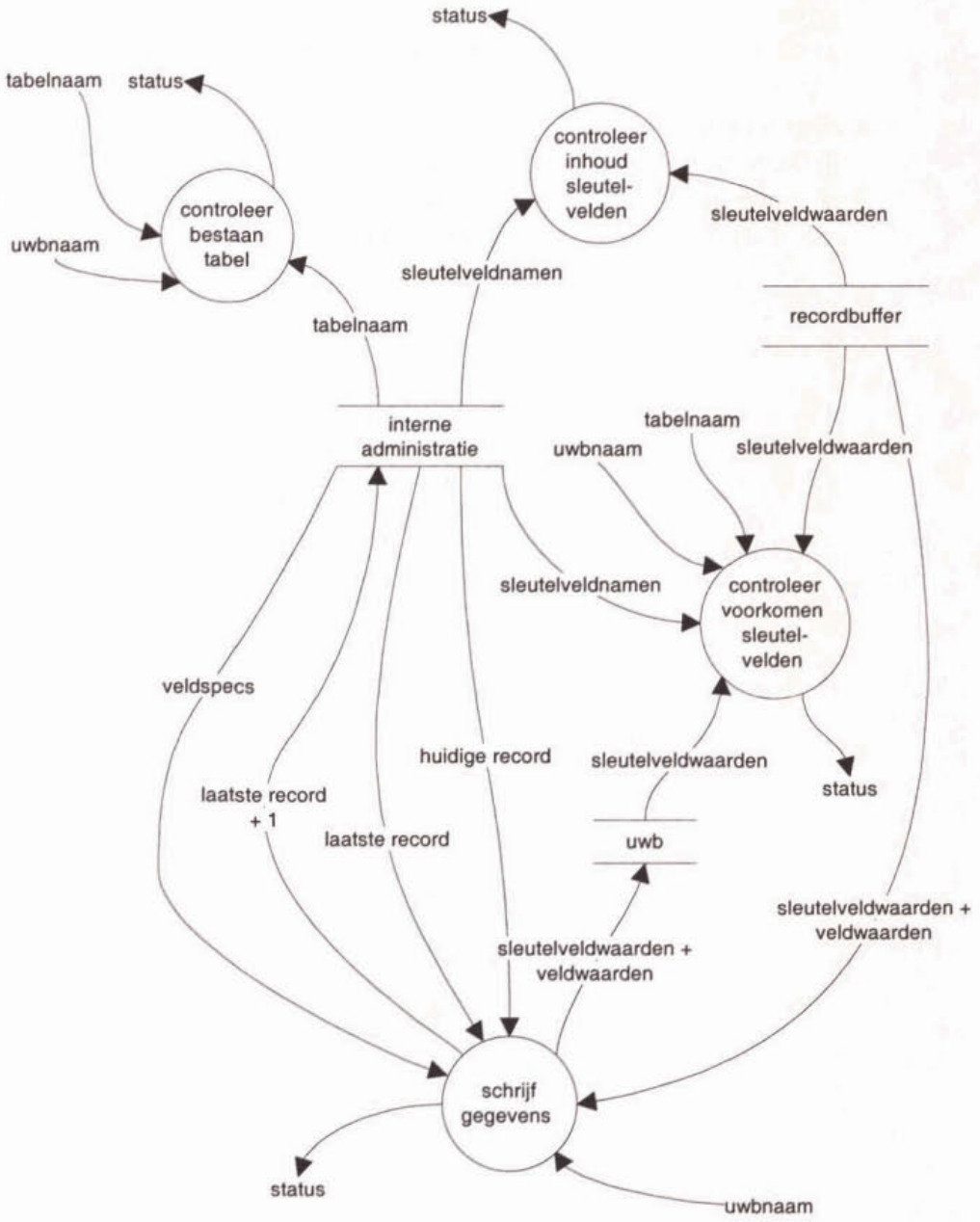
controleer veldnaam

Met deze functie wordt gecontroleerd of de veldnaam bestaat. Hiervoor wordt de veldnaam vergeleken met alle veldnamen in de *interne administratie* voor de betreffende tabel en uwb.

vul recordbuffer

Met deze functie wordt het *recordbuffer* gevuld.

3.3.9 Schrijven record



Figuur 11 Schrijven record

controleer bestaan tabel

Met deze functie wordt gecontroleerd of de tabelnaam bestaat op het *uwb*. Hiervoor wordt de tabelnaam vergeleken met alle tabelnamen in de *interne administratie* voor het betreffende uwb.

controleer inhoud sleutelvelden

Met deze functie gecontroleerd of de sleutelvelden zijn ingevuld.

controleer voorkomen sleutelvelden

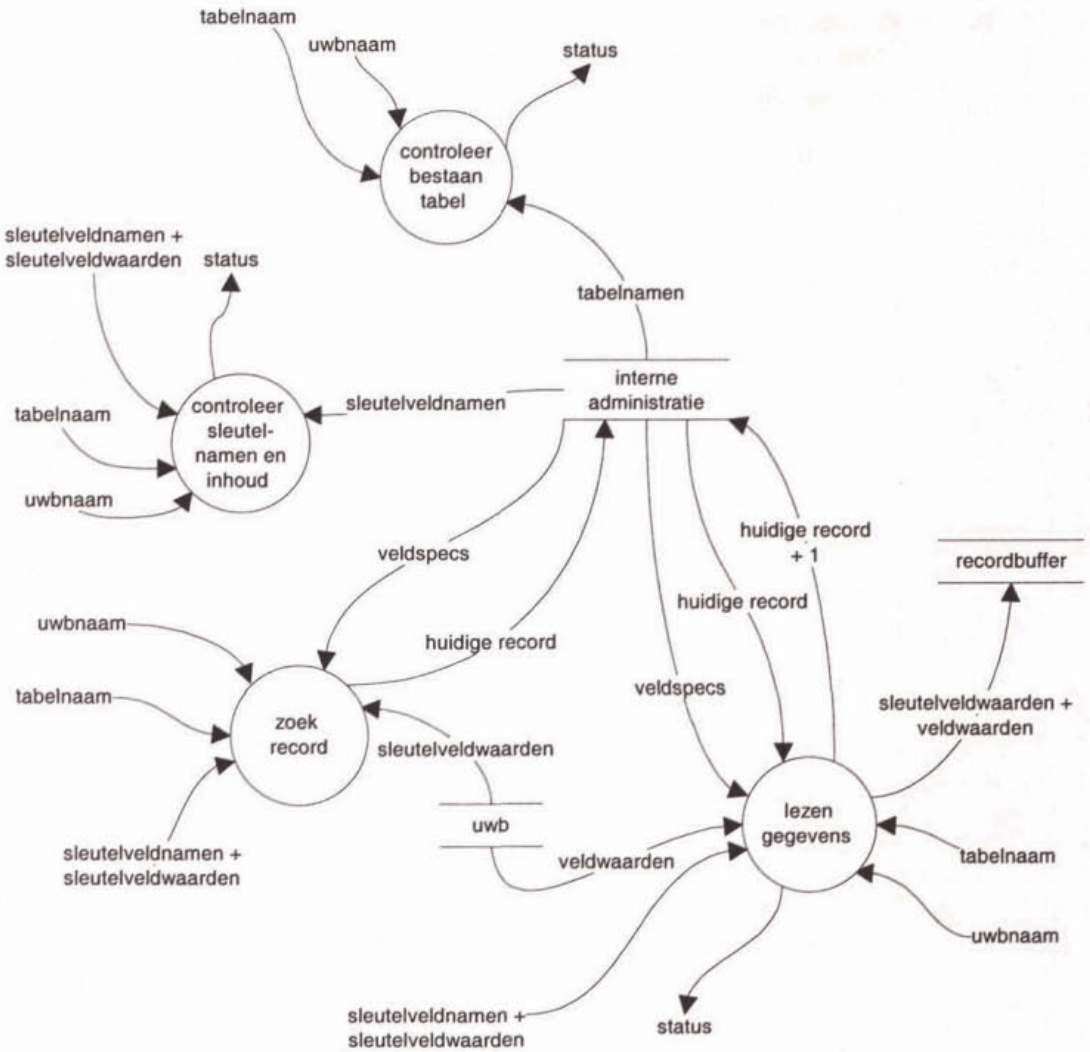
Met deze functie wordt gecontroleerd of de sleutelveldwaarden voorkomen op het *uwb*. Hiervoor worden de sleutelveldwaarden vergeleken met alle sleutelveldwaarden op het *uwb*.

Desgewenst kan deze functie worden uitgeschakeld (t.b.v. performance).

schrijf gegevens

Met deze functie worden gegevens naar het *uwb* geschreven. Controles worden nu niet meer uitgevoerd. Er kan aangegeven worden hoe het wegschrijven moet plaatsvinden; na het laatste record (= insert) of naar het huidige record (= update).

3.3.10 Lezen record



Figuur 12 Lezen record

controleer bestaan tabel

Met deze functie wordt gecontroleerd of de tabelnaam bestaat op het *uwb*. Hiervoor wordt de tabelnaam vergeleken met alle tabelnamen in de *interne administratie* voor het betreffende uwb.

controleer sleutelnamen en inhoud

Met deze functie worden de sleutelveldnamen gecontroleerd door deze te vergelijken met dat wat in de *interne administratie* staat voor de betreffende tabel en uwb.
Tevens wordt gecontroleerd of de sleutelveldwaarden zijn gevuld.
Desgewenst kan deze functie worden overgeslagen (bijv. t.b.v. performance).

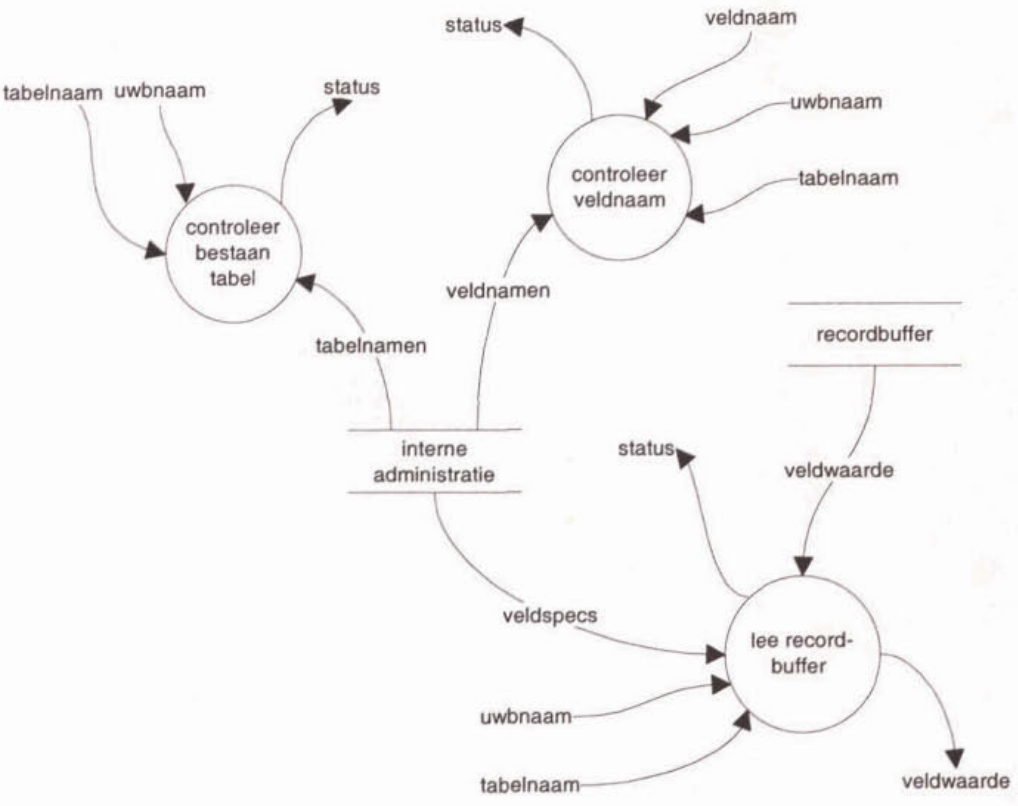
zoek record

Met deze functie wordt het record gezocht op basis van de sleutelveldwaarden. Desgewenst kan aangegeven worden dat niet gezocht hoeft te worden, hetgeen betekent dat bij het lezen automatisch het volgende record wordt gelezen.

lezen gegevens

Met deze functie worden de gegevens gelezen van het *uwb* en geplaatst in het *recordbuffer*.

3.3.11 Ophalen veld



Figuur 13 Ophalen veld

controleer bestaan tabel

Met deze functie wordt gecontroleerd of de tabelnaam bestaat op het *uwb*. Hiervoor wordt de tabelnaam vergeleken met alle tabelnamen in de *interne administratie* voor het betreffende uwb.

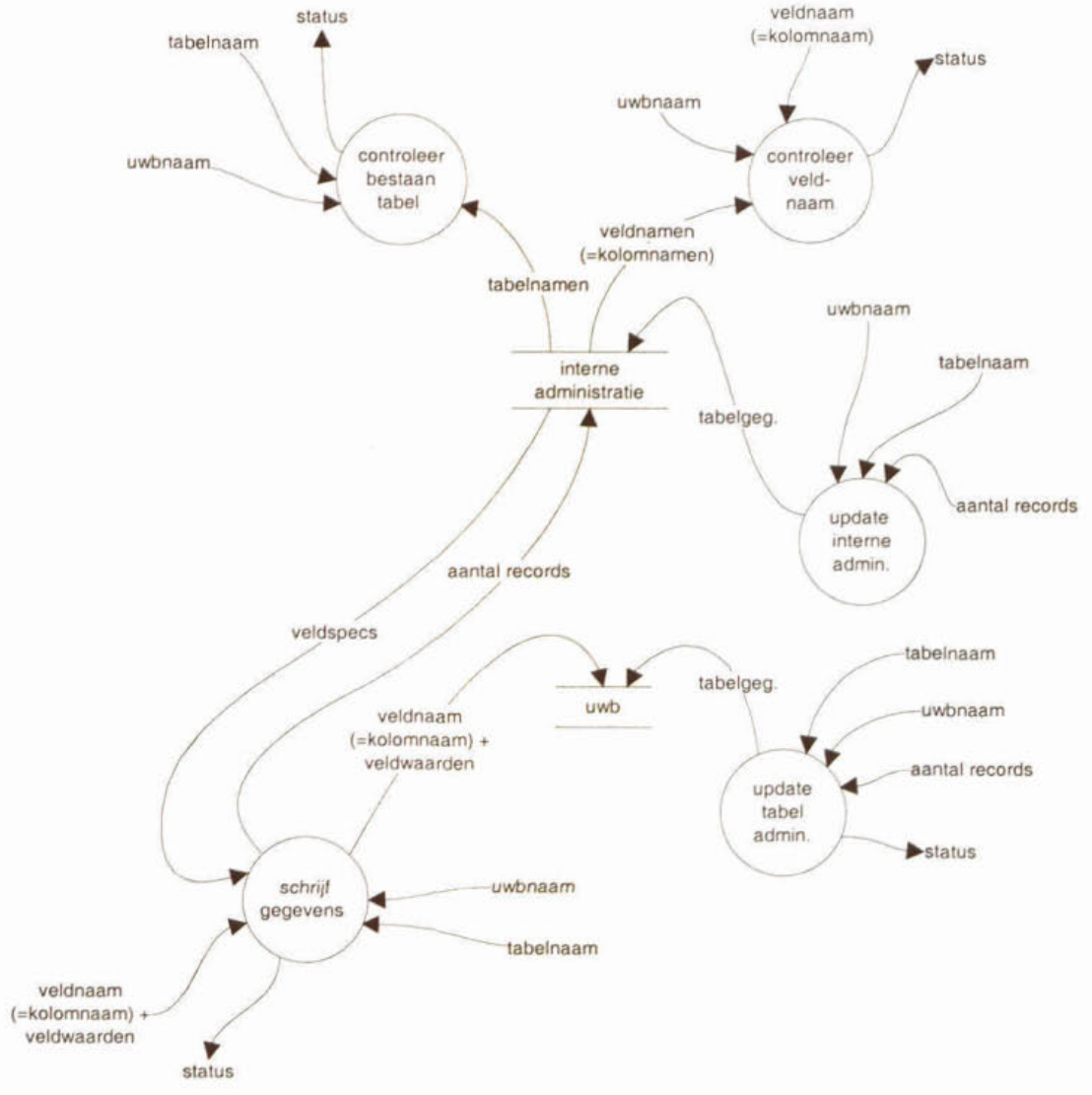
controleer veldnaam

Met deze functie wordt gecontroleerd of de veldnaam bestaat en correct is. Hiervoor wordt de veldnaam vergeleken met alle veldnamen in de *interne administratie* voor de betreffende tabel en uwb.

lees record buffer

Met deze functie wordt het gewenste veld gelezen uit het *recordbuffer*.

3.3.12 Schrijven kolom of reeks



Figuur 14 Schrijven kolom

controleer bestaan tabel

Met deze functie wordt gecontroleerd of de tabelnaam bestaat op het *uwb*. Hiervoor wordt de tabelnaam vergeleken met alle tabelnamen in de *interne administratie* voor het betreffende uwb.

controleer veldnaam

Met deze functie wordt gecontroleerd of de veldnaam bestaat en correct is. Hiervoor wordt de veldnaam vergeleken met alle veldnamen in de *interne administratie* voor de betreffende tabel en uwb.

update interne admin.

Met deze functie wordt de interne administratie bijgewerkt; d.w.z. de tabelnaam en de veldspecificaties worden toegevoegd.

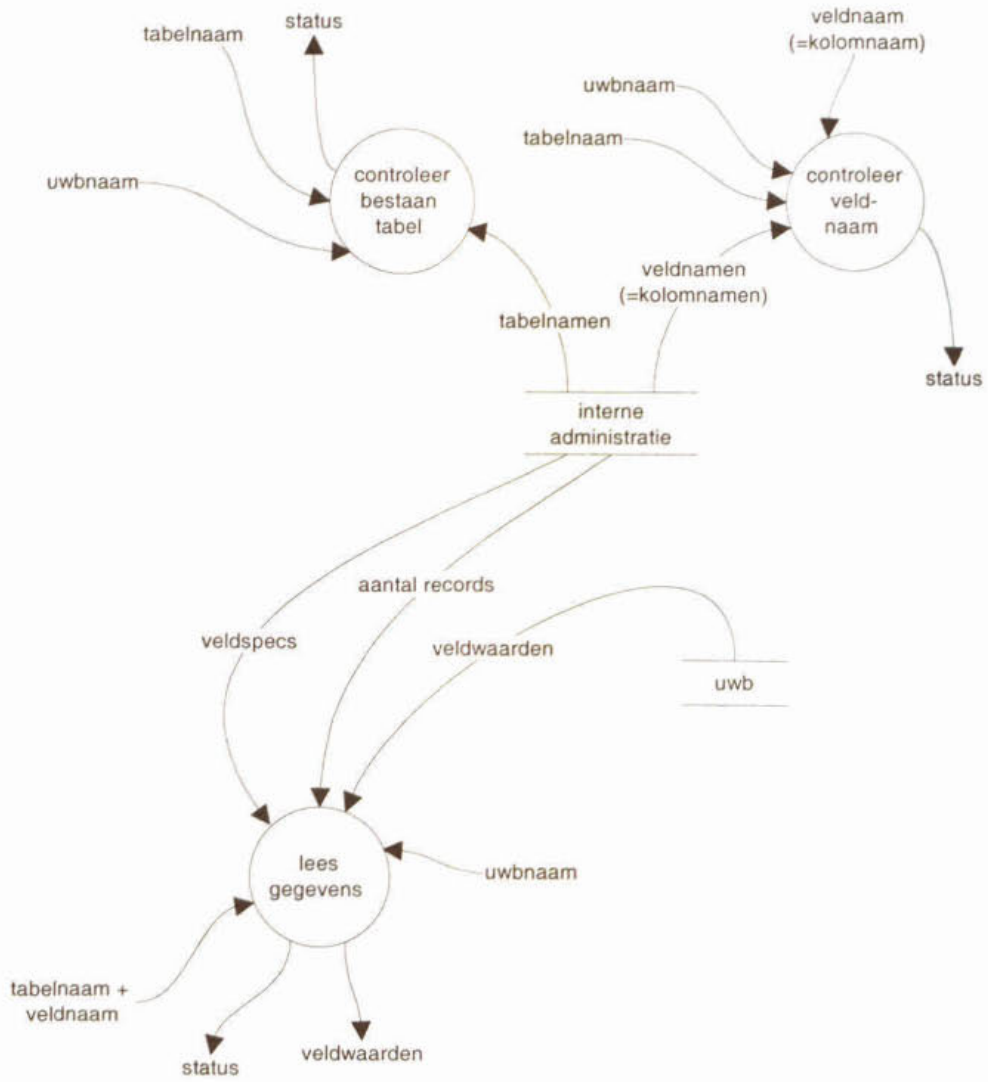
update tabel admin.

Met deze functie wordt de administratie tabel bijgewerkt; d.w.z. dat de tabelnaam wordt daarin weggeschreven.

schrijf gegevens

Met deze functie worden de veldwaarden rechtstreeks naar het *uwb* geschreven.

3.3.13 Lezen kolom of reeks



Figuur 15 Lezen kolom

controleer bestaan tabel

Met deze functie wordt gecontroleerd of de tabelnaam bestaat op het *uwb*. Hiervoor wordt de tabelnaam vergeleken met alle tabelnamen in de *interne administratie* voor het betreffende uwb.

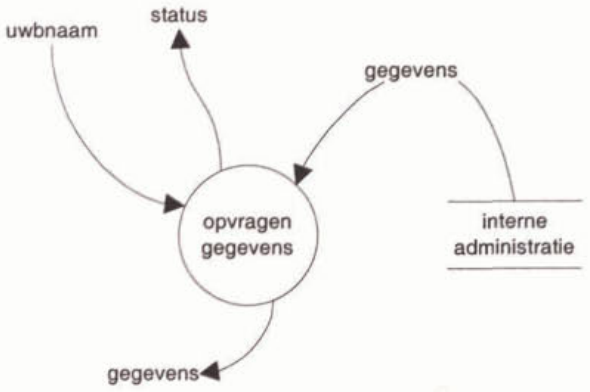
controleer veldnaam

Met deze functie wordt gecontroleerd of de veldnaam bestaat en correct is. Hiervoor wordt de veldnaam vergeleken met alle veldnamen in de *interne administratie* voor de betreffende tabel en uwb.

lees gegevens

Met deze functie worden de veldwaarden rechtstreeks van het *uwb* gelezen.

3.3.14 Opvragen gegevens



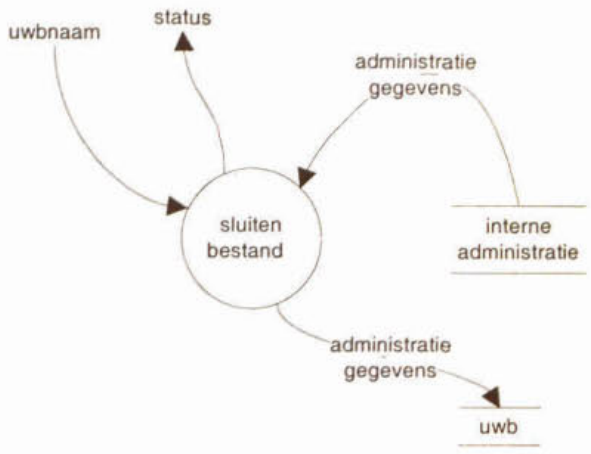
Figuur 16 Opvragen gegevens

De functie 'opvragen gegevens' bestaat uit de functies:

- vraag bestandsinformatie,
- vraag bestandsinhoud,
- vraag entiteittype informatie,
- vraag waardenreeksinformatie en
- vraag attribuut informatie.

Deze functies zijn in het vorige ontwerp gedefinieerd en zullen geen functionele wijzigingen ondergaan.

3.3.15 Sluiten bestand



Figuur 17 Sluiten bestand

sluiten bestand

Met deze functie worden de administratie gegevens zoals laatste record, datum, etc. opgehaald uit de *interne administratie* en weggeschreven naar het *uwb*.

A Additionele documenten

Hierachter zijn als bijlage toegevoegd twee documenten, welke aan de orde zijn geweest tijdens overleg met gebruikers over dit Functioneel Ontwerp.

DEFINITIEVE versie d.d. 22-04-1999

AFSPRAKENLIJST van de 2e vergadering van de Stekkerdoos Water versie 3 (voorheen ookwel on-line versie genoemd) op d.d. 2 februari 1999 te Utrecht.

Aanwezig
dr. ir. N. Brandenburg (NIG-TNO)
drs. M. Hogeweg (HKV)
ir. M.P.A.M. van de Looij (TU-Delft)
ing. T.W. van Urk (RIZA)
ing. J. Overmars (WL)
ing. C. v.d. Schelde (WL)
ir. L.R. Wentholt (STOWA)

Afwezig
ir. G. van Barneveld (RIZA)
ing. A.L. Berkhout (EcoSys)
ir. R. van den Boomen (Wi+Bo)
ir. F. Dirksen (RIZA)
ir. M. van Hengstum (MD)
ing. T.H. Holtel (BCC)
ir. L.A. Kaland (RWSR)
ir. J.J. Noort (Sepra)
ir. H. Reitsma (EDS)
ir. P.M. Eigeman (IKM)

1 Opening

Schriftelijke en/of mondelinge bijdragen zijn ontvangen van Van Hengstum (MD), Van Barneveld (CIW), Dirksen/Terveer (RIZA) via Van Urk, Kaland en Berkhout.

2 Presentatie en toelichting WL

Overmars ligt d.m.v. sheets het functioneel ontwerp toe.

3 Compilatie van de Discussie en afspraken n.a.v. de vorige 2e vergadering

NetCDF

WL is intern, maar met een lage prioriteit, aan het bezien of zij NetCDF in plaats van Nefis gaan gebruiken. Daar NetCDF public domain software is en de STOWA contractueel bij WL bedongen heeft dat zij de beschikking heeft over Nefis zal er qua beheersbaarheid niets veranderen.

Om het beheer en onderhoud van de Stekkerdoos goed te kunnen blijven vervullen wordt afgesproken de onderliggende software zo goed mogelijk te documenteren. De gebruiker van de Stekkerdoos dient niets te merken van het onderliggende script.

Afgesproken wordt op een zo laag mogelijk niveau lees- en schrijfacties te implementeren. De stekkerdoos dient een toegevoegde waarde op NetCDF te zijn en de gebruiker moet ADVENTUS entiteiten via de stekkerdoos kunnen aanbieden.

CIW-Infrastructuur

Tot het eind van het bouwtraject 'Stekkerdoos on-line' heeft de STOWA met WL een beheer en onderhoudscontract afgesloten. Vanaf 2000 zal de Stekkerdoos Water onderdeel gaan vormen van de CIW-Infrastructuur en dient daar ook de financiering geregeld te worden.

Voor de tussenperiode dient een nog een oplossing te worden gevonden. CIW zal de STOWA gaan verzoeken voor de periode tussen oplevering van de Stekkerdoos en het in CIW-kader in beheer komen van de Stekkerdoos het beheer en onderhoud te regelen.

Ontwikkelomgeving

Na geruime discussie wordt besloten te ontwikkelen voor een Windows95/NT omgeving en in Fortran. WL als uitgangspunt te hebben om zoveel mogelijk platform en systeemonafhankelijk te programmeren. In de documentatie zal expliciet worden vermeld waar de systeem-afhankelijk cals in de broncode staan vermeld.

WL dient de software wel met de gangbare en op de markt verkrijgbare Fortrancompiler te compileren. Verder dient de FORTRAN-software compatible te zijn met Fortran 90. Meerdere compilers maken een eenduidig beheer en onderhoud veel complexer. Afgesproken wordt een DLL uit te leveren gebaseerd op Digital Visual Fortran en indien men deze DLL niet gaat gebruiken zal men zelf zowel de Stekkerdooscode als de Nefiscode moeten gaan compileren. *Opmerking: men dient zich te realiseren dat dit uit oogpunt van beheer en onderhoud niet gewenst is.*

De DLL dient zodanig van opzet te zijn dat deze aan andere code's kan worden gehangen.

Upwards compatible

Afgesproken wordt om vanuit versie 2.0 een conversie te maken naar versie 3.0 (de on-line versie). Dit houdt in dat van versie 2.0 naar versie 3.0 de applicatie's (N.B. voorzover bekend spelen ze allemaal al in op de huidige ontwikkelingen) een éénmalige conversieslag dienen te maken, dit doordat in versie 3.0 op een andere manier naar de file's geschreven wordt. Vanaf versie 3.0 dient een stabiele situatie te worden gecreeërd met als uitgangspunt 'upwards compatible'.

De grootste meerwaarde van versie 3.0 ligt in de mogelijkheid van file-sharing (=meerdere uitwissleingsfile's) en performance verbetering.

Performance

Een uitgangspunt van dit project is het verbeteren van de performance, van te voren dient te worden vastgesteld welke orde-grootte deze dient te bedragen. De volgende vergadering moet deze worden vastgesteld. *Opmerking: hiertoe zal separaat door HKV en WL een gezamenlijk projectvoorstel worden opgesteld.*

File sharing (= meerdere uitwisselingsfile's)

Na een lange discussie wordt als compromis, en omdat nu niet te voorzien wat de behoefte is, afgesproken om het mogelijk te maken 25 bestanden te gelijk open te hebben. In principe is het mogelijk om geen bovengrens te stellen, er dient echter voor ieder file geheugen gealloceerd te worden.

Door gebruik te gaan maken van FORTRAN 90, wat in plaats van vaste geheugenallocatie gebruik maakt van dynamische geheugenallocatie, kan het systeem zelf gaan bepalen wat de grenzen zijn en vervalt de fysieke bovengrens van een bepaald aantal te gelijk open te hebben bestanden.

4 Compilatie van de Discussie en afspraken n.a.v. de laatste 3e vergadering

Waardenreeksen

Voorgesteld wordt om i.p.v. elke individuele waardenreeks een waardenreekstype tabel te maken is, zoals:

Veld	naam, status (conform ADVENTUS)
Veld	waardenblok (N-dimensionale array)

Overmars zal onderzoeken in hoeverre deze suggestie een verbetering zal opleveren.

Wegschrijven

Door per record i.p.v. per veld weg te schrijven lijkt een grote verbetering van de performance op te leveren. Dit kan worden gerealiseerd door eerst naar het geheugen weg te schrijven en het uiteindelijk wegschrijven van het record in één keer te doen.

Er valt nog meer performance te boeken indien je meerdere records kan bufferen. Dit kan door bij WRCRNT i.p.v. 1 attribuut een array van attributen weg te schrijven. Een probleem is wat te doen bij verschillende type attributen.

Naar aanleiding van de discussie of de recordbuffer niet per record, maar pas zodra de buffer vol is, weg moet schrijven lijkt het zinvol, om vanwege het onderscheid tussen on-line en off-line, om om een expliciete mogelijkheid van een extra call (flush) te vragen.

De mogelijkheid om kolomsgewijs weg te schrijven lijkt alleen bij een nieuw bestand voordelen op te leveren. Overmars zal deze suggestie nader uitwerken.

Een laatste suggestie is om niet standaard-Fortran functionaliteit te gebruiken om m.b.v. Microsoft Fortran Powerstation (STRUCTURE, RECORD) gegevenstypen in één lees- of schrijfopdracht naar of van file te gaan. Het zou zo mogelijk moeten zijn om een tabel in één keer in te lezen of weg te schrijven, hetgeen een grote interne file vereist. *Opmerking: Dit blijkt vanwege de structuur van Nefis niet mogelijk te zijn.*

Performance eis

De vergadering lijkt een performance van 10% langzamer dan NEFIS een redelijke eis. Een vergelijking van bv. 5000 meetwaarden intern BEVER en BEVER m.b.v. de Stekkerdoos zou een aardige indicatie moeten opleveren van wat de Stekkerdoos qua Performance uithaalt. Verder kan op de I/O van de datafile, de logfile en de foutfile ook het nodige worden verdiend.

Vanuit de RWSR wordt voorgesteld rekening te houden met reeksen van 30.000 tot 45.000 records.

5 Datum volgende vergadering, W.V.T.T.K. en sluiting

De volgende vergadering (derde) zal op aangeven van de uitvoerder plaatsvinden en via een datumbriefje bepaald worden.

Samenvatting van functies en wijzigingsvoorstellen versie 3 Stekkerdoos Water

Onderstaand een overzicht van de functies/routines van de Stekkerdoos Water. In versie 3 zijn t.o.v. versie 2 andere routinenamen gebruikt. Indien dezelfde routinenaam is gebruikt is de functionaliteit van versie 3 groter.
Bij het gebruik van de routines kan in versie 3 bij iedere tabel worden aangegeven op welk (dus meer dan 1) uitwisselingsbestand de gegevens moeten worden geschreven.
In versie 2.0 en 3.0 kan worden aangegeven dat ja/nee controle op sleutels moet plaatsvinden.
In versie 3.0 kan indien gewenst een vaste (bekende) lengte voor een tabel worden opgegeven. Beide laatste opties verhogen de performance.

In onderstaand overzicht wordt uitgegaan van een tabel die records (rijen) heeft. Ieder record bestaat uit velden. Een tabel kent ook kolommen (de velden van een tabel)
Het hier gebruikte begrip waardenreeksen is uitvoerig in het functioneel ontwerp (versie 1.0 van maart 199) uitgelegd.

Functie	**routines versie 2**	**routines versie 3**
Openen en sluiten uitwisselingsbestand	opnuwb clsuwb	opnuwb clsuwb
Maken van een tabel	makent	cretab
Schrijven van een waarde (=veld) in record	wr_key	wrvld (voor een veld) wrrec (voor het record)
Schrijven van veld in het actuele (current) record	wrcrnt	wrvld
Lees een waarde (=veld) van uit een record	rd_key	rdrec (voor het record) rdvld (voor een veld)
Lezen van veld uit het actuele (=current) record	rdcrnt	rdvld
Lees volgend record uit tabel	nxtobj	rdrec (met specifieke waarde voor een parameter)
Schrijven van een volledige kolom	--	wrkol
Lezen van een volledige kolom	--	rdkol
Maken van een waardenreeks	makwrd	crewrd
Schrijven 1-dimensionale waaardenreeks	wr1dwr	wrwrd
Lezen1-dimensionale waaardenreeks	rd1dwr	rdwrd
Schrijven meer-dimensionale waaardenreeks	wrndwr	wrwrd
Lezen meer-dimensionale waardenreeks	rdndwr	rdwrd
Opvraag functies voor: info over bestand info over de tabellen en waardenreeksen op bestand info over een specifieke tabel info over velden van een tabel info over specifieke wrdreeks	 inquwb inqcnt inqent inqatt inqwrd	 -- inquwb inqtbl aanwezig bij inqtbl inqwrd
Routine om een aantal zaken of waarden vooraf te zetten	--	setval

Onderstaand een opsomming van antwoorden op vragen en opmerkingen die door diverse personen zijn gesteld:

- Er is geprogrammeerd in Fortran90, waardoor de mogelijkheid ontstaat om als gebruiker aan te geven hoeveel uitwisselingsbestanden gelijktijdig open zijn.
- Er is gebruik gemaakt van Digital Visual Fortran (de opvolger van Microsoft Fortran Power Station).
- De broncode van de Stekkerdoos en van de Nefisfuncties is beschikbaar.
- De broncode is in standaard Fortran en kan daardoor naar een andere operatingsysteem met bijbehorende compiler worden overgebracht. Testen zal nodig blijven.
- Voor inspectie van het uitwisselingsbestand is een programma beschikbaar, de Viewer/Selector.
- Gemelde problemen met versie 2.0 routine NXTOBJ zijn opgelost.
- Een vaste plaats voor de file met foutmeldingen is niet meer nodig (was een beperking bij versie 2.00

Over het testen van de performance zijn met de STOWA afspraken gemaakt.

Stekkerdoos Water (versie 3.0)

Technisch Ontwerp

Inhoud

1 Inleiding 3

2 Opbouw van de code 4

2.1 Files met sourcecode 4

2.2 Generieke en specifieke routinenamen 5

2.3 Include en interface files 5

2.4 Stekkerdoos als DLL 6

2.5 Interne documentatie 6

2.6 Conventies voor de naamgeving 6

3 Structuur en gebruik globale variabelen 7

3.1 Structuur 7

3.2 Gebruik globale arrays en variabelen 9

3.2.1 mem.inc 9
3.2.2 afmeting.inc 9
3.2.3 index.inc 9

3.3 Foutenafhandeling 9

3.4 Initialiseren van variabelen 10

4 Interne administratie 11

4.1 Het idee achter interne administratie 11

4.2 Het idee achter recordbuffers 11

4.3 Datamodel van de interne administratie 12

4.3.1 Sessie 13
4.3.2 Uwb 14
4.3.3 Tabel 14
4.3.4 Veld 15

5 Opbouw NEFIS-file 16

5.1 Groepen voor de administratie 16

5.1.1 Groep voor de tabel-administratie 16
5.1.2 Groepen voor de veld-administratie 18
5.2 Datagroepen 21

6 Interne administratie versus NEFIS-file 23

6.1 Tabel-administratie 23
6.2 Veld-administratie 24

7 Uitwisselingsformaat 27

8 Gebruikte afkortingen 28

1 Inleiding

Dit Technisch Ontwerp is een uitwerking van het Functioneel Ontwerp en heeft als doelgroep de programmeur die de Stekkerdoos Water moet ontwikkelen en onderhouden. Applicatieprogrammeurs die de Stekkerdoos gaan toepassen kunnen evenueel dit document gebruiken om inzicht te krijgen in de achtergronden.

Er is een groepering van onderwerpen gemaakt in een aantal hoofdstukken. Daar staat tegenover dat veel zaken weer veel met elkaar te maken hebben. Het zal daarom voor u als lezer en gebruiker van dit document (soms) nodig zijn terug te grijpen op een vorig hoofdstuk of vooruit de kijken in een nog komend hoofdstuk. Anderzijds is getracht deze verwijzingen zoveel mogelijk bij de hoofdstukken aan te geven.

Bij eventuele discrepanties tussen het Functioneel Ontwerp en het Technisch Ontwerp, wordt het Technisch Ontwerp als definitief beschouwd.

In dit document is er van uitgegaan dat de programmeur van de Stekkerdoos Water software een grondige kennis heeft van de Nefis filestructuur, alsmede van het gebruik van de Nefis-functies.

2 Opbouw van de code

2.1 Files met sourcecode

De code is opgebouwd uit gebruikers-routines (zoals OPNUWB, CRETAB, WRREC, etc.,) en een aantal ondersteunende-routines (zoals: stringmanipulatie, error-afhandeling en zoek-functies).

Alle gebruikers-routines zijn in aparte files geplaatst. Bij de ondersteunende-routines heeft een clustering plaatsgevonden en bevatten de files meerdere routines.

De volgende files (met code) zijn beschikbaar.

			gebruikers routines	onder-steunende routines	functionaliteit
1	opnuwb	.f	x		openen uwb
2	clsuwb	.f	x		sluiten uwb
3	rduwf	.f		x	lezen uwf
4	cretab	.f	x		maken tabel
5	rdrec	.f	x		vullen recordbuffer
6	wrrec	.f	x		schrijven recordbuffer
7	rdvld	.f	x		lezen veld
8	wrvld	.f	x		schrijven veld
9	rwvld	.f		x	algemene routine voor lezen of schrijven veld
10	rdkol	.f	x		lezen kolom
11	wrkol	.f	x		schrijven kolom
19	rwkol	.f		x	algemene routine voor lezen of schrijven kolom
12	crewrd	.f	x		maken waardenreekstabel
13	rdwrd	.f	x		lezen waardenreeks
14	wrwrd	.f	x		schrijven waardenreeks
15	inq	.f	x		opvraagfuncties
16	str	.f		x	hulpfuncties voor stringmanipulatie
17	chk	.f		x	hulpfuncties voor controle
18	errlog	.f		x	error logging
20	init	.f		x	block data function
21	setval	.f	x		set-function

nb De nummers in de tabel worden gebruikt voor de fout- en meldingsnummers. Zo beginnen alle foutnummers van CRETAB met 4..., alle meldingsnummers beginnen met 41..
De teksten en nummers van de fouten en meldingen bevinden zich in de file: init.f

2.2 Generieke en specifieke routinenamen

Veel gebruikersroutines bevatten een variabele of array als parameter, waarmee de veldwaarde (inhoud) wordt geschreven of gelezen. Het type van deze parameter is afhankelijk van het gekozen Fortran datatype.
Er is voor gekozen om achter de generieke routinenaam uit de gebruikershandleidng een _s, _c , _i , _r of _d te plaatsen. Dat is voor de respectievelijke datatypen sleutel (=character), character, integer, real of double precision van velden, kolommen of waardenreeksattributen.
Op deze wijze wordt het datatype van alle routine parameters in de gehele code steeds correct 1 op 1 gehouden tot aan de Nefis functies PUTELT en GETELT. Deze twee functies kennen een numerieke parameter (buffer) voor de variabele of array met waarden die naar de file worden geschreven of er van af worden gelezen. Voor het geval dit een character parameter (buffer) is, zijn er de functies PUTELS en GETELS.

De genoemde files met de generieke naam bevatten dus steeds de routinenamen met de suffix. Deze laatsten vormen de (specifieke) gebruikersroutines.
Bij de gebruikte Fortran compiler is het dan weer mogelijk om via interface definities generieke routine-namen te maken met behulp van de specifieke routinenamen.

Voor het gebruik van de Stekkerdoos Water routinesuit de DLL in een andere programmeeromgeving zijn deze specifieke routinenamen ook nodig.

2.3 Include en interface files

Onderstaande tabel geeft overzicht van include files voor common en een specifieke file voor declaratie van alle administratieve arrays. Deze zaken worden verder uitgewerkt in hoofdstuk 3.

afmeting	.inc	common block met max. aantal tabellen en uwb's
errlog	.inc	common block met error/log informatie
index	.inc	common block met actuele uwb, tabel en veldnummer
mem	.inc	common block met alle administratie array's
decl	.inc	declaratie van alle administratie array's

Om het gebruik van verschillend type parameter (character, integer, real, double) in dezelfde functie-aanroep toe te staan zijn de volgende interface bestanden aanwezig. Daarin worden generieke routinenamen gedefinieerd. Ze zijn reeds gemaakt tijdens de ontwikkeling en testen van de software. De toepassing ligt bij het bouwen van applicaties en testprogramma's, waar generieke routinenamen kunnen worden toegepast.
(NB Voor de gebruiker/programmeur zullen er nog meer files en interfaces komen. Zij zijn geen onderdeel van het Technisch Ontwerp)

rdvld	.int	interface voor rdvld.f
wrvld	.int	interface voor wrvld.f
rdkol	.int	interface voor rdkol.f
wrkol	.int	interface voor wrkol.f

2.4 Stekkerdoos als DLL

De software van de Stekkerdoos Water wordt als DLL uitgeleverd. Dat heeft op de PC als groot voordeel dat andere talen als Visual Basic en Delphi er ook gebruik van kunnen maken.
Om die reden wordt er een compiler directive met DLLExport opgenomen in elke gebruikersroutine.

Verder wordt er van uit gegaan dat de programmeur van de Stekkerdoos Water software goed op de hoogte is van het gebruik en het maken van een DLL. Hij/zij moet ook de interfaces vanuit de DLL naar andere talen kunnen maken t.b.v. de gebruikers.

2.5 Interne documentatie

De code is als volgt intern gedocumenteerd:

- elke subroutine bevat een header waarin de gebruikte variabelen en parameters worden verklaard en
- elk block code (=ongeveer 10 regels) is voorzien van commentaar.

2.6 Conventies voor de naamgeving

De volgende conventie is aangehouden bij de namen van variabelen:

- alle array dimensies beginnen met: MX
- alle lokale loopvariabelen beginnen met: i_
- alle lokale variabelen (betrekking hebbend op aantallen) beginnen met: n_
- alle current variabelen beginnen met: j_
- alle variabelen voor hoogste of laatste beginnen met: n

De array dimensies zijn opgegeven/vastgelegd via het *parameter* statement van Fortran en worden geschreven met hoofdletters. Twee dimensies (voor aantal uitwisselingsbestanden en voor aantal tabellen op een bestand) kunnen door de gebruiker worden opgegeven via de routine SETVAL

De current variabelen beginnend met j_ hebben betrekking op de het gebruik van de interne administratie. Ze geven de plaats aan voor een uitwisselingsbestand (uwb), voor een tabel die gecreëerd is en voor een veld in een tabel. Dat worden dan respectievelijk j_uwb , j_tbl , j_vld. Zie ook Hoofdstuk 4 :Interne administratie".

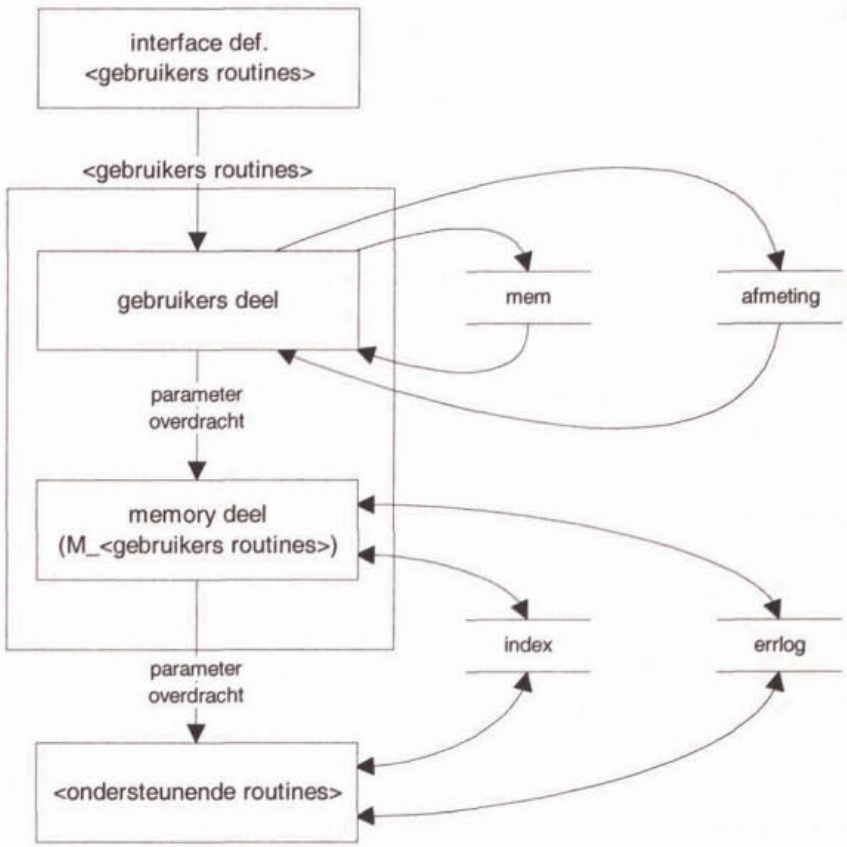
De naamgeving van routines is reeds vermeld in paragraaf 2.1.

Alle routine-parameters en lokale variabelen worden expliciet gedeclareerd. Ontbreekt een variabele in de declaratie dan is dat een common variabele of array.

3 Structuur en gebruik globale variabelen

3.1 Structuur

De volgende structuur is aangebracht in de code:



Nadere uitleg:

Van de gebruikersroutines worden interfaces gemaakt, zodat deze routines bruikbaar zijn vanuit Fortran, C, Visual Basic en Delphi. Sommige routines hebben een parameter die van een verschillend type (sleutel, character, integer, real double precision) is. Denk aan lezen en schrijven van velden, kolommen en attributen van waardenreeksen. Deze hebben specifieke namen door gebruik te maken van een suffix als _sc , _i , _r , _d. Bij sommige compilers is het mogelijk hiervoor weer generieke routine namen te definieren (o.a. in Fortran). Dat gebeurt via de interface definities van gebruikersroutines. Zie ook paragraaf 2.2 "Generiek en specifieke routinenamen".

Deze specifieke gebruikersroutines (met suffix) hebben de correcte parameters en dat is in de gehele code correct doorgevoerd tot en met de aanroep van de Nefis functies PUTELT en

PUTELS of GETELT of GETELS. Daarbij is tevens meegenomen het correcte gebruik van de interne recordbuffers voor de sleutel-, character-, integer-, real- en double precision velden.

Bij de Stekkerdoosroutines hebben we te maken met een interne administratie en record buffers, die op elk moment bij elke routine beschikbaar moeten zijn. Dat zijn de buffers *mem* en *afmeting* uit figuur hierboven. Ze zijn te beschouwen als globale variabelen en arrays. In Fortran kan dit worden geïmplementeerd met *common* of met *use* in Fortran90. Dit alles staat in de include files mem.inc en afmeting.inc en Hoofdstuk 4 beschrijft deze interne administratie.

Om nu voldoende flexibiliteit te hebben is er voor gekozen om de gebruikers-routines te laten bestaan uit een routine body waarin de twee onderdelen uit de figuur op de vorige pagina zijn te onderscheiden.
In de code van deze routines vindt u dan de volgende onderdelen:

- De noodzakelijke declaraties.
- Het includen van files voor de globale (=common of use) arrays en hun dimensies en andere globale variabelen. Dat zijn de files afmeting.inc en mem.inc;
- De aanroep van een nieuwe routine met dezelfde parameters, uitgebreid met alle globale interne administratie-arrays en hun afmetingen. Deze laatsten worden als parameters doorgegeven.
- De nieuwe routine start altijd met M_ en dan de gebruikers routinenaam, dus bijvoorbeeld OPNUWB en M_OPNUWB.

Voor deze M_ routines valt nog hetvolgende op te merken:

- Er is steeds sprake van dezelfde administratieve arrays en globale variabelen. Daarom is er voor gekozen deze onderdelen aan de aanroepende kant altijd en allemaal mee te geven naar iedere M_ routine. Dit deel van de parameterlist is daarmee altijd hetzelfde voor alle M_ routines.
- De declaraties voor de parameters die betrekking hebben op de interne administratie zijn op bovenstande wijze altijd identiek. Daarom is gekozen voor het 'includen' van de file decl.inc binnen de M_ routines. Deze file bevat alle genoemde declaraties.

De buffers *index* en *errlog* worden gebruikt voor de communicatie tussen de verschillende M_<gebruikers routines> en de <ondersteunende routines>. Er is één gebruikersroutine (SETVAL) die communiceert met errlog.
Deze buffers zijn geïmplementeerd middels common blocks en gedeclareerd in de include files index.inc en errorlog.inc.

De dynamische geheugendeclaratie voor de interne administratie is bekeken met het oog op complexiteit: het GW96 gegevensmodel en het opgeven van aantallen door de gebruiker. Baseren we ons het op GW96 datamodel, dan volgen hieruit maxima voor diverse zaken als aantal sleutel-, integer-, character-, real- en double precision velden en het maximum aantal velden in een entiteittype. De opzet van de interne administratie houdt hier rekening mee. De gebruiker kan deze details niet opgeven.

De gebruiker kan per applicatie wel de omvang van de interne administratie beïnvloeden via twee parameters. Dat zijn aantal openstaande uitwisselingsbestanden en het aantal tabellen dat naar een bestand wordt geschreven of er af gelezen.

3.2 Gebruik globale arrays en variabelen

Dit gebruik loopt via include files en maakt gebruik van Fortran *common* en *use*. Common wordt gebruikt voor statische variabelen en arrays en use voor dynamische arrays. De hieronder genoemde include files bevatten via commentaar voldoende uitleg.

3.2.1 mem.inc

Deze include file bevat alle array's en variabelen die nodig zijn om de interne administratie op te slaan. Zoals al eens eerder is opgemerkt wordt de interne administratie verder uitgelegd in Hoofdstuk 4. De maximale omvang van de arrays voor interne administratie wordt vastgelegd in de variabelen MXUWB en MXTBL, welke door de gebruiker kunnen worden opgegeven via routine SETVAL.

3.2.2 afmeting.inc

De file bevat een common declaratie voor de variabelen MXUWB en MXTBL.

3.2.3 index.inc

Alle namen van uitwisselingsbestanden, tabelnamen en waardenreeksnamen en veldnamen zijn opgeslagen in array's. Zodra via een gebruikersroutine een uwb-naam, tabelnaam of veldnaam wordt opgegeven, wordt m.b.v. de CHK...-routine de index bepaald in deze arrays.

Het common block *index* bevat de indexvariabelen (j_uwb, j_tbl, j_vld) van het actuele uitwisselingsbestand, tabel en veld.

3.3 Foutenafhandeling

De foutenafhandeling loopt via een foutnummer en een tekstuele uitleg. Nummers en fouten zijn opgeslagen in globale arrays. Deze arrays worden gedeclareerd in de file errlog.inc en in de file staat voldoende commentaar. Routines voor foutenafhandeling staan in de file errlog.f. Fouten worden naar een file geschreven en de gebruiker kan aangeven via routine SETVAL of hij ja/nee foutmeldingen wil zien.

3.4 Initialiseren van variabelen

Hierover zijn de volgende twee opmerkingen te maken:

- De globale variabelen die in common staan en een initiële waarde nodig hebben, worden via *block data* van Fortran geïnitialiseerd. Het is de file init.f
- De record buffers (ook globale variabelen) zijn dus bij aanvang geïnitialiseerd en worden vervolgens ja/nee opnieuwe geïnitialiseerd na het schrijven van een record. Het ja/nee kan door de gebruikers bij de routine voor schrijven record worden aangegeven.

4 Interne administratie

Om een maximale performance te kunnen behalen, worden de lees- en schrijfacties van en naar file tot het minimum beperkt. Dat heeft tot gevolg dat er een interne administratie is opgezet alsmede buffers voor sleutel-, integer-, real-, character- en double precisionvelden voor de attributen van de tabellen (=entiteitypen), die de gebruiker naar het uitwisselings-bestand schrijft. Beide onderdelen worden hierna verder uitgewerkt.

4.1 Het idee achter interne administratie

Maximale performance wordt bereikt als de de volledige administratie in het geheugen wordt gehouden; de zogenaamde interne administratie van de Stekkerdoos.
Hiermee wordt bijgehouden:

- welke uwb's (uitwisselingsbestanden) er in gebruik zijn,
- welke tabellen er bij ieder uitwisselingsbestand horen,
- welke velden horen bij de tabellen,
- wat zijn het huidige record en het laatste reocord in een tabel,
- allerhande 'handige' gegevens per tabel.

Het idee is om bij het lezen en schrijven niet steeds de NEFIS-file te hoeven raadplegen. Dit is niet alleen tijdrovend (INQuire functie van Nefis), maar ook erg inefficiënt. Een eenmaal gedefinieerde en gebruikte tabel of veld zal nl. niet wijzigen (in de loop der tijd).

4.2 Het idee achter recordbuffers

Als tweede stap is het noodzakelijk om de velden (=attributen) van een tabel (=entiteittype) per datatype te groeperen.
Per tabel ontstaan er derhalve recordbuffers en ook afzonderlijkeNefis groepen voor:

- sleutel-velden
- character-velden
- integer-velden
- real-velden
- double precision-velden

Ieder veld komt overeen met een plaats in de recordbuffer en een apart element in de groep (zie ook hoofdstuk 5).

Door deze groepering is het mogelijk maximaal gebruik te maken van NEFIS-functionaliteit omdat met één lees- of schrijfopdracht (GETELT of PUTELT) alle velden van het zelfde datatype gelezen of geschreven kunnen worden (door gebruik te maken van de '*'-optie).
Het volledig uitlezen van een record is dus maximaal 5 lees opdrachten, ondanks dat er misschien wel 40 velden (=attributen) zijn.

Voor de verschillende datatypen worden 5 verschillende recordbuffers bijgehouden. De Fortran namen zijn:

- recbfs voor de sleutel-velden
- recbfc voor de character-velden
- recbfi voor de integer-velden
- recbfr voor de real-velden
- recbfd voor de double precision-velden

In de recordbuffers *recbfs* en *recbfc* worden alle character-velden achter elkaar geplaatst; deze recordbuffers bevatten dus één lange characterstring.

De recordbuffers recbfi, recbfr en recbfd bevatten 1d-array's van het juiste datatype.

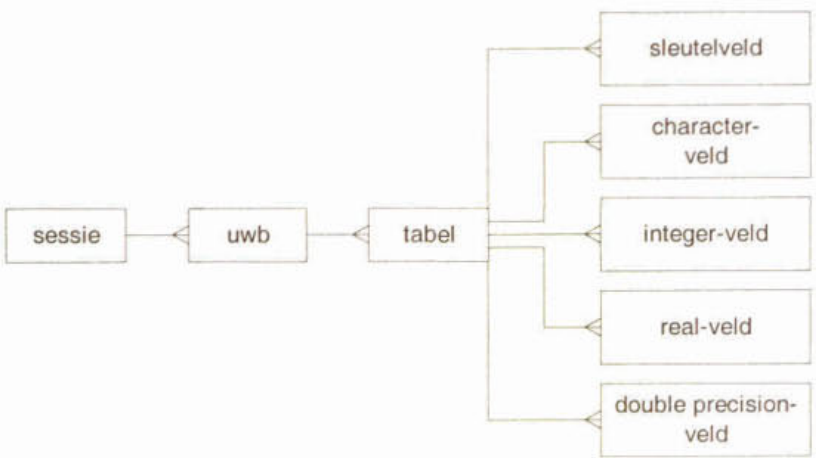
Er is via de interne administratie bekend welk veld, welke index in de recordbuffer heeft, of zijn plaats in de stringbuffer. Zo kan de waarde van een veld direct uit het geheugen worden gehaald of daarin worden geplaatst.

4.3 Datamodel van de interne administratie

(Data)modellerings gezien is er de volgende relatie:

Bij een applicatie (=sessie) met de gebruikersfuncties komen deze relaties naar voren omdat hierbij moet worden opgegeven welke uwb wordt gebruikt, welke tabel het betreft en van de tabel over welk veld het gaat.

Als de velden worden uitgesplitst naar de verschillende datatypen ontstaat het volgende diagram:



Deze relatie (of gelaagdheid) komt in de Stekkerdoos software tot uitdrukking bij de dimensionering van de diverse administratie array's. Daar wordt in de volgende paragrafen verder op ingegaan.

Binnen het gegeven eenvoudige datamodel kan gezegd worden:

- Op het nivo van *sessie* heeft de interne administratie een afmeting die onafhankelijk is van MXUWB (=aantal uwb's), MXTBL (=aantal tabellen) of MXVLD(=aantal velden).
- Op het niveau van *uwb* hebben de array's een afmeting: MXUWB.
- Op het niveau van *tabel* hebben de array's een afmeting: MXTBL*MXUWB.
- Op het niveau van *veld* hebben de array's een afmeting: MXVLD*MXTBL*MXUWB.

4.3.1 Sessie

Entiteittype *sessie* bevat algemene informatie die iets zegt over de sessie (=executie van een gebruikers-applicatie met de Stekkerdoosroutines).

De attributen (=variabelen) van *sessie* hebben de volgende kenmerken:

attribuut of variabele	omschrijving	datatype	dimensie
lopene	vlag om aan te geven of de error-file al is geopend	logical	1
lopenl	vlag om aan te geven of de log-file al is geopend	logical	1
lunerr	lun van de error-file	integer	1
lunlog	lun van de log-file	integer	1
nuwb	aantal uitwisselingsbestanden	integer	1
lerr	vlag voor wel dan niet schrijven van fouten naar error-file	logical	1
llog	vlag voor wel dan niet schrijven van meldingen naar log-file	logical	1
nrerr	nummer van de fout-melding	integer	1
nrlog	nummer van de log-melding	integer	1
txterr	fout-meldingen	char*76	MXERR
txtlog	log-meldingen	char*76	MXLOG
buffer	buffer voor ophalen sleutel-velden	char*1	MXBUF
recbfs	recordbuffer voor sleutel-velden	char*1	MXsLEN
recbfc	recordbuffer voor character-velden	char*1	MXcLEN
recbfi	recordbuffer voor integer-velden	integer	MXiLEN
recbfr	recordbuffer voor real-velden	real	MXrLEN
recbfd	recordbuffer voor double precision-velden	double precision	MXdLEN
uindex	user-index (NEFIS)	integer	3

De implementatie van entiteit *sessie* in de Fortran code zijn variabelen en arrays met de naam, zoals die gegeven is in kolom attribuut en de dimensie van de kolom dimensie. Zie ook de include files mem.inc en errlog.inc.

4.3.2 Uwb

De entiteit *uwb* zegt iets over de uwb's (=uitwisselingsbestanden) zelf. Deze gegevens worden niet op het bestand zelf bewaard, omdat ze per sessie kunnen verschillen. De gegevens worden tijdens run-time (OPNUWB, CRETAB) bijgehouden.

De attributen (=variabelen) van *uwb* hebben de volgende kenmerken:

attribuut of variabele	omschrijving	datatype	dimensie
uwbnm	uwb namen	char*256	MXUWB
lrwuwb	vlag voor schrijfpermissie	logical	MXUWB
deffds	buffer voor .def-file	integer	2993,MXUWB
datfds	buffer voor .dat-file	integer	999,MXUWB
lunlog	lun log-file	integer	MXUWB
ntbl	aantal tabellen (per uwb)	integer	MXUWB

De implementatie van entiteit *uwb* in de Fortran code zijn arrays met de naam, zoals die gegeven is in de kolom attribuut en met de dimensie van de kolom dimensie. Zie ook de include file mem.inc

4.3.3 Tabel

Entiteittype *tabel* bevat informatie die betrekking heeft op de tabellen (=entiteittypen) welke de gebruiker uitwisseld via een uwb.

De attributen (=variabelen) van *tabel* hebben de volgende kenmerken:

attribuut of variabele	omschrijving	datatype	dimensie
tblnm	tabelnamen	char*13	MXTBL,MXUWB
tbllen	characterlengte van de tabelnaam	integer	MXTBL,MXUWB
nvld	totaal aantal velden	integer	MXTBL,MXUWB
nvlds	aantal sleutel-velden	integer	MXTBL,MXUWB
nvldc	aantal character-velden	integer	MXTBL,MXUWB
nvldi	aantal integer-velden	integer	MXTBL,MXUWB
nvldr	aantal real-velden	integer	MXTBL,MXUWB
nvldd	aantal double precision-velden	integer	MXTBL,MXUWB
tbldef	tabeldefinitie	char*13	MXTBL,MXUWB
tbldim	dimensies	integer	MXTBL,MXUWB,5
currec	huidige record	integer	MXTBL,MXUWB
lstrec	laatste recordnummer	integer	MXTBL,MXUWB

De implementatie van entiteit *tabel* in de Fortran code zijn arrays met de naam, zoals die gegeven is in de kolom attribuut en met de dimensie van de kolom dimensie. Zie ook de file mem.inc.

Omdat de gegevens niet wijzigen (per sessie) worden deze gegevens opgeslagen op het uitwisselingsbestand (= uwb). Tijdens run-time (OPNUWB, CRETAB) worden de gegevens van de uwb gelezen of aangevuld.

4.3.4 Veld

Entitieittype *veld* bevat informatie die betrekking heeft op de velden van de gebruikers tabellen (=entiteittypen).

De attributen (=variabelen) van *veld* hebben de volgende kenmerken:

attribuut of variabele	omschrijving	datatype	dimensie
vldnms	veldnamen van de sleutel-velden	char*8	MXsVLD,MXTBL,MXUWB
vldnmc	veldnamen van de character-velden	char*8	MXcVLD,MXTBL,MXUWB
vldnmi	veldnamen van de integer-velden	char*8	MXiVLD,MXTBL,MXUWB
vldnmr	veldnamen van de real-velden	char*8	MXrVLD,MXTBL,MXUWB
vldnmd	veldnamen van de double precision-velden	char*8	MXdVLD,MXTBL,MXUWB
vldnm	alle veldnamen	char*8	MXVLD,MXTBL,MXUWB
tbldnm	groepnaam van de data-tabel	char*16	MXVLD,MXTBL,MXUWB
elmidx	indexnummer van een veld in een element	integer	MXVLD,MXTBL,MXUWB
lstvld	laatste indexnummer van dat veld in een groep	integer	MXVLD,MXTBL,MXUWB
vldlns	veldlengte van de sleutel-velden	integer	MXsVLD,MXTBL,MXUWB
vldlnc	veldlengte van de character-velden	integer	MXcVLD,MXTBL,MXUWB

De implementatie van entiteittype *tabel* in de Fortran code zijn arrays met de naam, zoals die gegeven is in de kolom attribuut en met de dimensie van de kolom dimensie. Zie ook de file mem.inc

Omdat de gegevens niet wijzigen (per sessie) worden deze gegevens opgeslagen op de uwb. Tijdens run-time (OPNUWB, CRETAB) worden de gegevens van de uwb gelezen en/of aangevuld.

5 Opbouw NEFIS-file

Op de NEFIS-file zijn 2 soorten groepen aanwezig, te weten:

- groepen die informatie van de interne administratie bevattten
- data groepen van de uit te wisselen tabellen

Voor beide soorten groepen wordt in de onderstaande paragrafen verder uitgewerkt hoe ze naar het uitwisselingsbestand (=de Nefis file) worden geschreven.

5.1 Groepen voor de administratie

De groepen voor de administratie zijn onderverdeeld in:

- één groep voor de (algemene) administratie, de entiteit *tabel*,
- voor iedere gecreëerde gebruikerstabel: 5 groepen voor de veldadministratie.

5.1.1 Groep voor de tabel-administratie

De groep voor de tabel-administratie is bedoeld om informatie over aantallen vast te leggen, die vervolgens worden gebruikt voor administratie over de velden behorenden bij een specifieke gebruikerstabel. Het gaat hier om de opslag van gegevens in een groep die behoren tot de entiteit *tabel*.

De groep voor de tabel-administratie heeft de volgende structuur:

		dimensie		
groepnaam	TABEL	*		
celnaam	TABEL	nvt	*type*	*omschrijving*
elementnamen	TABELNAAM	1	char*13	tabelnaam
	NR_S_VELDEN	1	integer	aantal sleutel-velden
	NR_C_VELDEN	1	integer	aantal character-velden
	NR_I_VELDEN	1	integer	aantal integer-velden
	NR_R_VELDEN	1	integer	aantal real-velden
	NR_D_VELDEN	1	integer	aantal double precision-velden
	TABELDEF	1	char*13	tabeldefinitie (zie nadere verklaring)
	TABELDIM	5	integer	dimensies van de datagroep en elementen (zie nadere verklaring)
	LAATSTE_RECORD	1	integer	laatst geschreven record (zie nadere verklaring)
	BILATERAAL	1	char*1	vlag om aan te geven of een entiteittype een bilaterale afspraak is

U vindt hierin gemakkelijk de overeenkomst tussen elementnamen en de attribuutnamen (arrays) van de entiteittype *tabel* uit paragraaf 4.3.3. Zie verder ook hoofdstuk 6 voor relatie tussen deze groep en entiteittype *tabel*.

De afmeting (dimensie) van deze groep wordt bepaald door het aantal tabellen dat wordt gebruikt voor lezen en schrijven van/naar een uwb (vrijde groepsdimensie van Nefis).

Hiermee is het geheel flexibel en dat is nodig omdat het aantal tabellen dat naar een uwb wordt geschreven onbekend is.

De dimensie (=aantal gebruikerstabellen) van de arrays, die de attributen van entiteittype *tabel* vormen, wordt hier dus opgevangen in de groepsdimensie. De dimensie van het aantal uwb's wordt opgevangen door per uwb te werken.

Op iedere uwb is deze groep aanwezig.

Nadere verklaring

In de nu volgende uitleg speelt het (kleine) verschil en de overeenkomst tussen entiteittypen en waardenreeksen. Dat is uitvoerig beschreven in het Functioneel Ontwerp.
Samengevat: ook waardenreeksen kunnen worden beschouwd als entiteittypen, maar de attributen zijn anders dan bij 'gewone' entiteittypen, het zijn waardenverzamelingen (arrays).
Van 'gewone' entiteittypen ontstaan slechts één tabel en daarmee één groep op de Nefis file. Van een waardenreeks als entiteittype kunnen meerdere waardenreeksen worden gecreëerd. De gebruiker geeft de specifieke/unieke naam op. Je kunt dit een nieuwe tabel noemen en het geeft steeds een nieuwe groep op de Nefis file.
Voor de eenduidigheid binnen de interne administratie en de code wordt hier voor beide 'soorten' entitiiteittypen de term tabelnaam en tabeldef gebruikt. Zie ook het voorbeeld.

5.1.1.1 TABELNAAM en TABELDEF

TABELDEF is de naam van het entiteittype zoals die voorkomt op het uwf (file met uitwisselingsformaat)
TABELNAAM is naam van de tabel zoals die op de NEFIS-file staat. In geval van een 'normale' tabel zijn TABELNAAM en TABELDEF gelijk.
In geval van waardenreeksen verschillen deze. Een waardenreeks-definitie, zoals TIJDREEKS uit het voorbeeld, wordt meerdere keren gebruikt voor bijvoorbeeld T1 en T2 en dat worden dus tabelnamen.

5.1.1.2 TABELDIM (1d-array met 5 elementen)

TABELDIM(1) = groepsdimensie

Als gewerkt wordt met een vrije-dimensie is deze 0, anders bevat deze de waarde zoals opgegeven bij de functie CRETAB

Bij iedere nieuwe waardenreeks via functie CREWRD wordt de (vaste) lengte van de waardenreeks opgegeven.

TABELDIM(2..5) = dimensies van de velden van de datagroep in het geval van een waardenreeks.

In het geval van een 'normale' tabel is deze per definitie 1,0,0,0.

Voor waardenreeksen bevat deze de afmetingen/dimensies van de attributen, dus zijn er 1 tot 4 getallen ingevuld.

5.1.1.3 LAATSTE RECORD

Voor tabellen van 'gewone' entiteittypen staat hier het nummer van het laatste record. Voor waardenreeksen, die veld voor veld (dus eigenlijk als een soort kolom) worden geschreven staat hier de lengte van de waardenreeks onder de voorwaarde dat alle velden (=attributen) zijn geschreven. In andere gevallen staat hier een 0.

Als bij 'gewone' tabellen kolom voor kolom wordt geschreven (en naar dat groepstype) is voor de Stekkerdoos niet duidelijk of alle kolommen geschreven zijn, ze even lang zijn en alle groepstypen zijn gebruikt. De gebruiker kan dit nl. voor een beperkt aantal kolommen hebben gedaan. Hiermee kunnen we dan niet spreken van een laatste record voor de gehele tabel. Zijn alle kolommen wel geschreven, dan is dit wel het geval en heeft nummer van laatste record wel zin. In de code wordt aldus getest en geprogrammeerd.
Als record voor record naar file is geschreven kunnen later wel bepaalde velden als kolom worden teruggelezen.
Er kan nog opgemerkt worden dat door elkaar gebruiken van tabel-functies voor kolommen en records vermeden moet worden.

5.1.1.4 VOORBEELD

TABEL (groepnaam)
TABEL (celnaam)
De cellen zijn alsvolgt:

TABELNAAM	NR_S_VELDEN	NR_C_VELDEN	NR_I_VELDEN	NR_R_VELDEN	NR_D_VELDEN	TABELDEF	TABELDIM	LAATSTE_RECORD	BILATERAAL
KST	2	37	2	1	0	KST	0,1,0,0,0	35	N
OWA	2	10	2	7	0	OWA	20,1,0,0,0	18	N
T1	0	1	1	1	0	TIJDREEKS	500, 1,0,0,0	500	J
T2	0	1	1	1	0	TIJDREEKS	200, 1,0,0,0	200	J
GRID1	0	0	0	2		GRID	1, 127,213,0,0	1	J
POLY1	0	0	0	2	0	POLYGOON	103, 1,0,0,0	103	J

nb Op logisch niveau ligt het voor de hand om dimensies van kolom 'tabeldim' op nul te zetten, als ze niet worden gebruikt. In de code worden ze echter op 1 gezet om gemakkelijker te kunnen vermenigvuldigen voor de grootte van de Nefis-buffer voor PUTELT en GETELT

5.1.2 Groepen voor de veld-administratie

Administratieve informatie over velden is opgeslagen in de entiteit *veld*, zie ook paragraaf 4.3.4. De attributen van *veld* worden naar de Nefis-file geschreven en er van teruggelezen. Dat gebeurt per Nefis-file en per gebruikerstabel die naar deze file geschreven wordt. Er zijn per gebruikerstabel (=entiteittype) maximaal 5 groepen en elke groep bevat de veld (=attribuut) gegevens van een bepaald datatype.

Zo ontstaan er groepen voor:

- de sleutel-velden
- de character-velden
- de integer-velden
- de real-velden
- de double precision-velden

De groepnaam wordt afgeleid van de TABELNAAM. De afzonderlijke groepnamen heten:

<TABELNAAM>_S	voor sleutel-velden
<TABELNAAM>_C	voor character-velden
<TABELNAAM>_I	voor integer-velden
<TABELNAAM>_R	voor real-velden
<TABELNAAM>_D	voor double precision-velden

Met deze aanpak worden de attributen van *veld* (paragraaf 4.3.2) in slechts 1 dimensie naar de Nefis-file geschreven. De dimensie van het aantal tabellen wordt opgevangen via de groepsnaam zoals hierboven is aangegeven. De dimensie van het aantal uwb's wordt opgevangen door deze groepen naar de juiste uwb te schrijven. Hier ontstaat ook maximale flexibiliteit ten aanzien van de grootte van de elementen voor de veld-administratie. Elementen (behalve EENHEID) bestaan uit 1d arrays, die een onderdeel zijn van de 3d arrays als genoemd in paragraaf 4.3.4 "Veld". Omdat de afmetingen bekend zijn, is gekozen voor elementen als arrays met vaste lengte. Zie verder ook nog Hoofdstuk 6 voor een relatie met attributen van entiteittype *veld* en de Nefisgroepen.

De groep voor veld-administratie heeft de volgende structuur:

		dimensie		
groepnaam	<TABELDEF>_<datatype>	1		
celnaam	VELDSPECS	nvt	*type*	*omschrijving*
elementnamen	VELDNAAM	(zie	char*8	veldnaam
	OMSCHRIJVING	nadere	char*64	omschrijving
	EINDPOSITIE	verklaring)	integer	eindpositie in de characterstring (zie nadere verklaring)
	EENHEID	1	char*16	eenheid
	LAATSTE_VELD	1	integer	rangnr. van laatste gebruikte veld

Nadere verklaring

5.1.2.1 DIMENSIE VAN DE ELEMENTEN

Deze dimensie is vastgelegd in het element: NR_x_VELDEN van de groep: TABEL. Dat geldt tevens voor het element LAATSTE_VELD.

5.1.2.2 VELDNAAM

VELDNAAM is een element dat alle veldnamen bevat van één datatype, zoals die voorkomt op de file met uitwisselingsformaat (uwf) en ook in de interne administratie. Het opslaan gebeurt in de zelfde volgorde zoals ze voorkomen in de administratie (en ook op de uwf). Er wordt onderscheid gemaakt in datatype, maar dat zit ook reeds in de naam van de

groep en er is dus een relatie tussen de naam van de groep en het datatype van en aantal velden in het element VELDNAAM. Zie verder ook nog hoofdstuk 6.

5.1.2.3 EINDPOSITIE

De EINDPOSITIE bevat de totale characterlengte (=eigen lengte + alle voorgaande velden). Dit is zo gekozen omdat alle character-velden en sleutelvelden als één string in het recordbuffer worden geplaatst.
Voor de groepen van het type <TABELNAAM>_I, <TABELNAAM>_R en <TABELNAAM>_D worden deze elementen niet gebruikt.

5.1.2.4 OMSCHRIJVING en EENHEID

OMSCHRIJVING en EENHEID zijn de nadere verklaringen van de velden zoals ze voorkomen op de uwf. Deze gegevens worden niet opgenomen in de interne administratie maar alleen opgenomen op het uwb.

5.1.2.5 LAATSTE_VELD

Van ieder veld wordt bijgehouden zijn laatste 'record' in de groep, wanneer schrijven per kolom gebeurt. Dat geldt tevens voor het schrijven van de velden (=attributen) van waardenreeksen. Kijk ook nog onder 5.1.1 "Groep voor tabel-administratie" en daar bij nadere verklaring van LAATSTE_RECORD.

nb Van de entiteiten waarvoor een bepaald datatype niet bestaat wordt geen groep voor veld-administratie gemaakt. De groep: <TABELNAAM>_D zal dus in de meeste gevallen ontbreken.

nb Van de waardenreeksen bestaan maximaal een set van 4 groepen voor de veld-administratie, omdat voor waardenreeksen de groep met sleutelvelden ontbreekt.

Voor waardenreeksen geldt (uitgaande het voorbeeld bij de tabel-administratie) dat er één veld-administratie groep TIJDREEKS_R bestaat (en niet 2 groepen: T1_R en T2_R).

5.2 Datagroepen

De informatie van de gebruikerstabellen wordt in datagroepen opgeslagen op een uwb. Om een optimale performance te krijgen met zo weinig mogelijk lees- en schrijfacties is er voor gekozen om de velden van eenzelfde type (=eenzelfde type attributen van enititeittypen) samen te voegen binnen een groep. Zo ontstaan er per tabel (=entiteittype) maximaal 5 data groepen. Er zijn groepen voor:

- sleutel-velden
- character-velden
- integer-velden
- real-velden
- double precision-velden

Voor de waardenreeeksen gelt ook dat ze beschouwd kunnen worden als entiteittypen met attributen. Zie hiervoor ook de uitleg van waardenreeksen in het functioneel ontwerp.
Voor de interne administratie en het schrijven naar Nefis-files is van deze aanpak zoveel mogelijk gebruik van gemaakt in de code van de Stekkerdoos. Er zijn twee kleine verschillen ten aanzien van de tabelnaam en de tabeldimensie.
Van iedere waardenreeks als entiteittype is een unieke waardenreeks te creëren en de attributen zijn geen enkelvoudige waarden, maar een waardenverzameling. Hierover is in meer detail geschreven in paragraaf 5.1.1.

De groepnaam wordt afgeleid van de TABELNAAM in verband met gebruik van waardenreeksen. De afzonderlijke groepnamen heten:

<TABELNAAM>_SD	voor sleutel-velden
<TABELNAAM>_CD	voor character-velden
<TABELNAAM>_ID	voor integer-velden
<TABELNAAM>_RD	voor real-velden
<TABELNAAM>_DD	voor double precision-velden

Ieder van de 5 data-groepen heeft de volgende structuur, waarbij <datatype> is een S , C , I , R of D:

		dimensie
groepnaam	<TABELDEF>_<datatype>D	(zie nadere verklaring)
celnaam	<TABELDEF>_<datatype>D	nvt
elementnamen	veld_1 (zie veld_2 nadere veld_3 verklaring) .. .. veld_n	(zie nadere verklaring)

Bij de waardenreeksen ontbreekt de groep voor sleutelvelden. Verder geldt dat een datagroep zoals hierboven genoemd alleen ontstaat als het desbetreffende type veld ook aanwezig is in de tabel (of entiteittype).

Nadere verklaring

5.2.1.1 DIMENSIE VAN DE GROEP

De dimensie van de groep is vastgelegd in het element TABELDIM(1) van de groep TABEL en ook aan de daarmee gerelateerde arrays van de interne administratie in de code. Zowel een vrije dimensie als een vaste dimensie is mogelijk.

5.2.1.2 DIMENSIE VAN DE ELEMENTEN

De dimensie van de elementen is vastgelegd in het element TABELDIM(2..5), van de groep TABEL en ook aan de daarmee gerelateerde arrays van de interne administratie in de code. Voor 'normale' tabellen is dat per definitie 1 en alleen TABELDIM(2) wordt gebruikt. Voor 'waardenreeks' tabellen is dit de afmeting/dimensie van de attributen en wordt dit opgegeven tot maximaal een aantal van 4 in TABELDIM(2.....5). Zie ook paragraaf 5.1.1 "Groep voor de tabel-administratie" onder het onderdeel 'nadere uitleg' het voorbeeld.

5.2.1.3 ELEMENTNAMEN

Ieder veld wordt een eigen element in de groep en daarmee bevat de groep evenveel elementen als er velden zijn in de overeenkomstige groep voor veld-administratie. Het aantal is vastgelegd in het element NR_x_VELDEN van de groep TABEL.

6 Interne administratie versus NEFIS-file

In hoofdstuk 4 is o.a. gesproken over entiteittypen *tabel* en *veld* en afbeelding van hun attributen in Fortran arrays. Daarna is in hoofdstuk 5 besproken hoe deze attributen van *tabel* en *veld* (dus de Fortran arrays) naar de Nefis-file worden geschreven.

Onderstaand wordt in tabelvorm weeergegeven de relatie tussen elementnamen van de groep en van de attributen (= array namen) van de genoemde entiteittypen. In hoofdstuk 5 is in detail uitgelegd hoe de informatie uit deze attributen/arrays in de Nefis-groep en elementen wordt weggeschreven.

6.1 Tabel-administratie

De gegevens opgeslagen in de tabel: TABEL (op de uwb) zijn gekoppeld aan de entiteit: *tabel* (van de interne administratie).

Groepnaam: TABEL	Entiteittype *tabel*
elementnaam	attribuut (array naam)
TABELNAAM	tblnm
-	tbllen
-	nvld
NR_S_VELDEN	nvlds
NR_C_VELDEN	nvldc
NR_I_VELDEN	nvldi
NR_R_VELDEN	nvldr
NR_D_VELDEN	nvldd
TABELDEF	tbldef
TABELDIM	tbldim
-	currec
LAATSTE_RECORD	lstrec
BILATERAAL	-

De variabele <tbllen> wordt niet opgeslagen op de uwb; deze wordt run-time afgeleid.
De variabele <currec> wordt niet opgeslagen op de uwb; deze wordt run-time bijgehouden.

Het element <BILATERAAL> wordt niet in geheugen opgeslagen, omdat deze slechts zeer sporadisch nodig is (INQTBL) en niet nodig is voor de verwerking van de gegevens.

6.2 Veld-administratie

De informatie opgeslagen in de iterne administratie volgens entiteittype *veld* wordt mett een reeks van groepen op een Nefis-file geschreven. Voor iedere gebruikerstabel of waardenreeks ontstaan een aantal groepen volgens datatype van de velden van zo'n tabel of waardenreeks. De groepsnaam wordt samengesteld uit de naam gegeven door het veld TABELNAAM uit de groep tabel. De inhoud van het element VELDNAAM is afhankelijk van het datatype van het veld. Het gaat hier om de Fortran array namen: vldnms, vldnmc, vldnmi, vldnmr en vldnmd. Voor sleutel en character datatypen zijn ook nog de Fortran arrays vldlns en vldlnc nodig.
Welke datatypen werkelijk aanwezig zijn wordt door de elementen NR_x_VELDEN uit groep tabel gegeven, nl. wanneer ze ongelijk nul zijn.

Aldus ontstaan de volgende groepen voor de Nefis file.

groepnaam: <TABELNAAM>_S	Entiteittype *veld*
elementnaam	attribuut (aaray naam)
VELDNAAM	vldnms
OMSCHRIJVING	-
EINDPOSITIE	vldlns
EENHEID	-
-	elmidx
-	vldnm
-	tbldnm
LAATSTE_VELD	lstvld

De attributen (Fortran arrays) <elmidx>, <vldnm> en <tbldnm> worden niet opgeslagen op de uwb; deze wordt run-time afgeleid.

De elementen <OMSCHRIJVING> en <EENHEID> worden niet in geheugen opgeslagen, omdat ze slechts sporadisch nodig zijn via routine INQVLD) en niet nodig zijn voor de verwerking van de gegevens.

Voor de andere datatypen volgt het overzicht op de volgende pagina's.

groepnaam: <TABELNAAM>_C	Entiteittype *veld*
elementnaam	attribuut (aaray naam)
VELDNAAM	vldnmc
OMSCHRIJVING	-
EINDPOSITIE	vldlnc
EENHEID	-
-	elmidx
-	vldnm
-	tbldnm
LAATSTE_VELD	lstvld

groepnaam: <TABELNAAM>_I	Entiteittype *veld*
elementnaam	attribuut (aaray naam)
VELDNAAM	vldnmi
OMSCHRIJVING	-
EINDPOSITIE	-
EENHEID	-
-	elmidx
-	vldnm
-	tbldnm
LAATSTE_VELD	lstvld

groepnaam: <TABELNAAM>_R	Entiteittype *veld*
elementnaam	attribuut (aaray naam)
VELDNAAM	vldnmr
OMSCHRIJVING	-
EINDPOSITIE	-
EENHEID	-
-	elmidx
-	vldnm
-	tbldnm
LAATSTE_VELD	lstvld

groepnaam: <TABELNAAM>_D	Entiteittype *veld*
elementnaam	attribuut (aaray naam)
VELDNAAM	vldnmd
OMSCHRIJVING	-
EINDPOSITIE	-
EENHEID	-
-	elmidx
-	vldnm
-	tbldnm
LAATSTE_VELD	lstvld

7 Uitwisselingsformaat

De uwf heeft de volgende structuur en format.

Het filetype is: Direct Access, Formatted, Recordlengte = 91.

De file is opgedeeld in 3 blokken:

Blok 1 bavat het aantal entiteiten dat beschreven is.
Blok 2 bevat een beschrijving van de entiteiten.
Blok 3 bevat een beschrijving van de attributen.

Blok	*recnr*	*format*	*inhoud*
1	1	i5	aantal entiteiten (=normale tabellen en waardenreekstabellen)
2	2 t/m aantal entiteiten	a12,6i5,a1,5i5	totaal aantal attributen van de entiteit aantal sleutel attributen aantal character attributen aantal integer attributen aantal real attributen aantal double precision attributen indicator voor bilaterale entiteit (J/N) dimensies(4) van de elementen igv waardenreeks recordnummer van waaraf de attrubuut gegevens staan (nb vanaf dat record staan achtereenvolgens de sleutel-, character-, integer-, real- en double precision-attributen)
3	n	a8,i3,a64,a16	attribuutnaam lengte igv sleutel- of character attribuut omschrijving eenheid

8 Gebruikte afkortingen

uwb = uitwisselingsbestand
uwf = uitwisselingsformaat

lun = logical unit voor gebruik in Fortran open, read, write, close statements