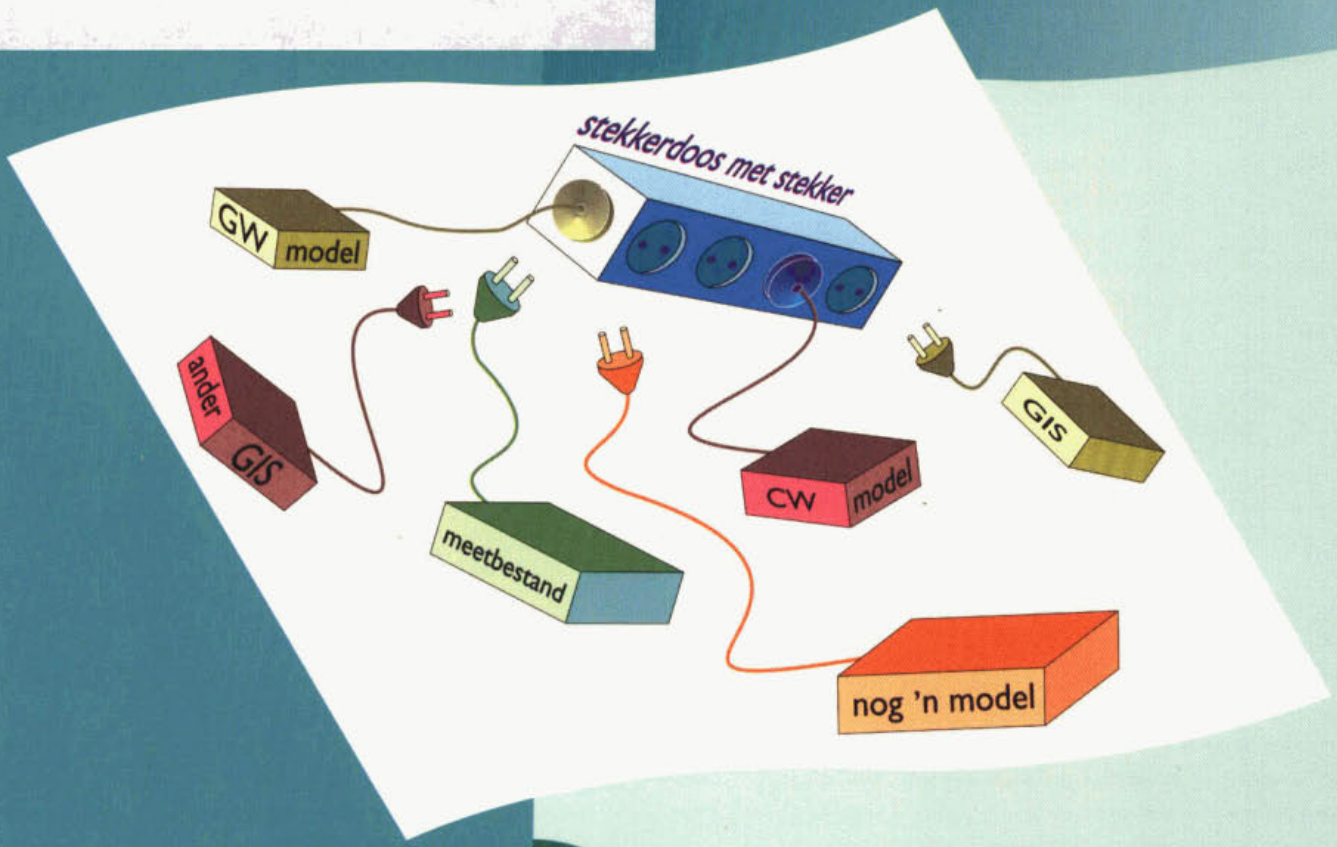
stowa

Stichting Toegepast Onderzoek Waterbeheer

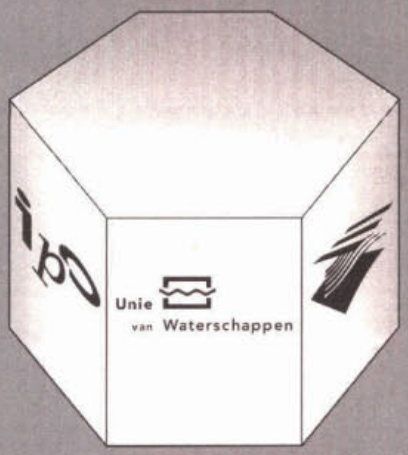
STEKKERDOOS WATER

- *Gebruikershandleiding*
- *Functioneel ontwerp*

1998-04_stekkerdoos-water-gebruikershandleiding



98 04



stowa

Stichting Toegepast Onderzoek Waterbeheer

STEKKERDOOS WATER

- Gebruikershandleiding

- Functioneel ontwerp

98 04

Publicaties en het publicatie-overzicht van de STOWA kunt u uitsluitend bestellen bij:

Hageman Verpakkers BV

Postbus 281

2700 AC Zoetermeer

o.v.v. ISBN- of bestelnummer en een duidelijk afleveradres.

ISBN 90.5773.017.0

Ten geleide

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 en voor het uitwisselen van waardereeksen, zoals meetreeksen en rekenresultaten. Het gekozen opslagformaat biedt tevens de generieke functionaliteit voor het opslaan van schematisaties, modelgegevens en modelresultaten van één of meerdimensionale waterbewegingsmodellen, grondwatermodellen en waterkwaliteitsmodellen. Het is daarom zeer geschikt voor 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 Stekkerdoos Water is dus een bibliotheek welke gebruikt kan worden voor het realiseren van Stekkers. De gebruikers van de Stekkerdoos Water zijn de programmeurs die stekkers bouwen.
Voor de Stekkerdoos Water is gekozen voor een oplossing op basis van de NEFIS-bibliotheek voor een optimale uitwisseling van waardereeksen (meetgegevens, modeldata etc.), die ook de gegevensuitwisseling van NEN 1878 en NEN 3610 gegevens ondersteunt. Vanwege o.a. het bestuurlijk draagvlak van de NEN 1878 bij voornamelijk uitwisseling van administratieve en geografische objectgegevens zal in een parallel spoor worden onderzocht in hoeverre een twee lagen stekkerdoos een zinvolle oplossing zal genereren.
De voordelen van de Stekkerdoos Water zijn een vermindering van complexiteit en kosten voor de organisaties en een verbetering van de kwaliteit van de gegevensuitwisseling door uniformering middels de Stekkerdoos Water. Doordat de Stekkerdoos Water datgene wat universeel is bevat, kunnen organisaties die willen uitwisselen volstaan met de bouw van relatief eenvoudige stekkers. De gebruiker dient er attent op te zijn dat bij uitwisseling zowel een exportstekker als een importstekker aanwezig dient te zijn dan wel gebouwd moet worden.
Op de voorjaar 1998 uit te brengen ADVENTUS CD-ROM zal eveneens de Stekkerdoos Water worden meegeleverd, naast de gebruikershandleiding en het functioneel ontwerp zal daarop ook het technisch ontwerp worden bijgevoegd. Deze CD-ROM zal ook de bijbehorende software bevatten.
Dit STOWA-onderzoek is mede gefinancieerd door IPO, RIZA en Unie van Waterschappen. De werkzaamheden zijn uitgevoerd door WL | delft hydraulics, met als projectleider de heer R.F.M. Bouwmeister en als projectmedewerkers ing. F.A. Bouma en ing. J. Overmars. In de begeleidingscommissie hadden zitting: met als voorzitter ing. H. ter Veen (Unie van Waterschappen), dhr. J.C.M. de Beer (Hoogheemraadschap West-Brabant), dr. N.P. Brandenburg (NITG), ir. F. Dirksen (RIZA), ing. J. Hendriksma (RIZA), ir. P. Mulder (Provincie Zuid-Holland), dhr. H. Krijnsen (Waterschap Friesland), ing. F.J.M. Latjes (Waterschap Hollands Kroon), dhr. P. de Leeuw (Gis-Zes), dhr. J.H. van Oogen (Ravi) en ir. L.R. Wentholt (STOWA).

Utrecht, mei 1998

De directeur van de STOWA

drs. J.F. Noorthoorn van der Kruijff

Stekkerdoos
Water

Gebruikers Handleiding

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 Functiebeschrijving 1-3
1.4.1 Uitwisseling van GW'96 geclassificeerde gegevens 1-4
1.4.2 Uitwisseling aan de hand van bilaterale afspraken 1-5
1.4.3 Uitwisseling van waardereeksen 1-5
1.4.4 Waardereeksen in de Stekkerdoos Water 1-6
2 Installatieprocedure 2-1
2.1 Systeemeisen 2-1
2.2 Platforms en portabiliteit 2-1
2.3 Overzicht files op diskette 2-1
2.3.1 Bibliotheek bestanden 2-1
2.3.2 Include/header files 2-2
2.3.3 Datafiles 2-2
2.3.4 Executables en SDW-scripts 2-2
2.3.5 Documentatie 2-2
2.3.6 Stuurbestand GW96 2-2
2.3.7 C-interface 2-3
2.3.8 Voorbeelden 2-3
2.4 Installatie-instructie 2-3
2.5 Programmeur instructies 2-3
3 Systeembeschrijving 3-4
3.1 GW'96 + waardereeksen 3-5
3.2 Stuurbestandgenerator 3-5
3.3 Stuurbestand 3-6
3.4 Bilateraal bestand 3-6
3.5 NEFIS-definitie-generator (CREDEF) 3-7
3.6 NEFIS-definitiebestand 3-8
3.7 Header-files 3-9
3.8 Stekkerdoos-bibliotheek 3-9
3.8.1 Algemeen 3-9
3.8.2 Performance 3-10
3.8.3 Functies 3-10
3.8.4 Overzicht van de functies 3-12
3.9 Applicatie/stekker 3-13
3.9.1 Algemene opmerkingen 3-13
3.9.2 Het schrijven van een uitwisselingsbestand 3-13
3.9.3 Het lezen van een uitwisselingsbestand 3-14
4 Ontwerp- en implementatiebeslissingen 4-1
5 DLL 5-1
5.1 Algemeen 5-1
5.2 Microsoft Fortran 5-1
5.3 Microsoft C/C++ 5-1
5.4 Visual Basic 5-2
6 Detailbeschrijving bibliotheekfuncties 6-3
6.1 Inleiding 6-3
6.2 Performance 6-3
6.3 Openen en sluiten van het uitwisselingsbestand 6-5
6.3.1 Open uitwisselingsbestand 6-5
6.3.2 Sluit het uitwisselingsbestand 6-6
6.4 GW'96 uitwisseling 6-7
6.4.1 Creëer een entiteittype 6-7
6.4.2 Schrijf waarde met sleutel 6-8

6.4.3 Schrijf een waarde naar het laatst geselecteerde record ... 6-10
6.4.4 Lees een waarde van het eerst geselecteerde record ... 6-11
6.4.5 Lees nog een waarde van het laatst geselecteerde record ... 6-13
6.4.6 Selecteer het eerstvolgende object ... 6-14
6.5 Uitwisseling van waardereeksen ... 6-15
6.5.1 Creëer een waardereeks ... 6-15
6.5.2 Schrijf een één dimensionale waardereeks ... 6-17
6.5.3 Schrijf een n-dimensionale waardereeks ... 6-18
6.5.4 Lees een één dimensionale waardereeks ... 6-19
6.5.5 Lees een n-dimensionale waardereeks ... 6-20
6.6 Functies om informatie op te vragen ... 6-21
6.6.1 Vraag informatie van het uitwisselingsbestand ... 6-21
6.6.2 Vraag bestandsinhoud ... 6-23
6.6.3 Vraag entiteittype informatie ... 6-24
6.6.4 Vraag waardereeksinformatie ... 6-25
6.6.5 Vraag attribuut informatie ... 6-27
6.7 Beschrijving constanten ... 6-28
7 Detailbeschrijving stuurbestanden ... 7-1
7.1 GW-classificatie ... 7-1
7.1.1 Deel 1: Administratieve gegevens ... 7-1
7.1.2 Deel 2: Gegevenselementen ... 7-1
7.1.3 Deel 3: Entiteittypen ... 7-2
7.2 Bilaterale afspraken ... 7-3
7.2.1 Algemeen ... 7-3
7.2.2 Deel 1: Administratieve gegevens ... 7-3
7.2.3 Deel 2: Gegevenselementen ... 7-3
7.2.4 Deel 3: Entiteittypen ... 7-4
7.3 Voorbeeld stuurbestand ... 7-5
8 Overzicht foutmeldingen ... 8-1
8.1 Foutmeldingen CREDEF ... 8-1
8.2 Waarschuwingen Stekkerdoos Water ... 8-3
8.3 Foutmeldingen Stekkerdoos Water ... 8-4
8.4 NEFIS foutmeldingen ... 8-7
9 Voorbeelden voor stekkers ... 9-1
9.1 Fortran stekker ... 9-1
9.1.1 Voorbeeld programma 1 ... 9-1
9.1.2 Uitvoer programma 1 ... 9-3
9.1.3 Voorbeeld programma 2 ... 9-4
9.1.4 Uitvoer programma 2 ... 9-7
9.1.5 Voorbeeld programma 3 ... 9-8
9.1.6 Uitvoer programma 3 ... 9-11
9.2 C-Stekker ... 9-12
9.2.1 Sources ... 9-12
9.2.2 Programma uitvoer ... 9-18
10 Referenties ... 10-1

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 en gelezen naar een uitwisselingsbestand. Dit uitwisselingsbestand is bedoeld voor de overdracht van gegevens tussen applicatieprogramma's onderling, maar ook voor de overdracht van gegevens tussen organisaties.

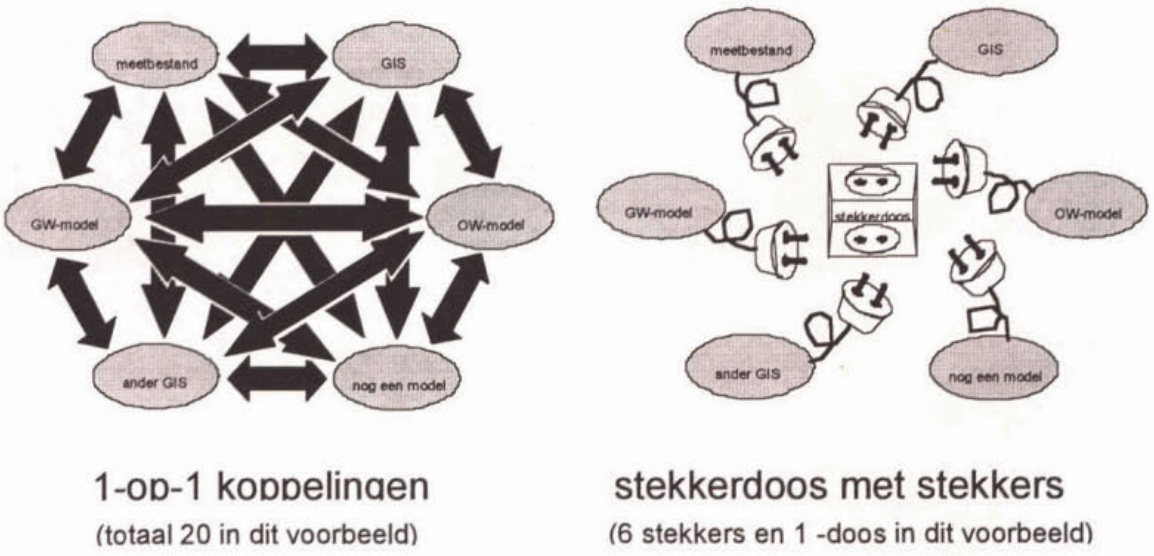
Daarnaast biedt de Stekkerdoos Water mogelijkheden voor het uitwisselen van waardereeksen zoals meetreeksen en rekenresultaten via hetzelfde uitwisselingsbestand.

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

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 (GW'96) [REF-1]. 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

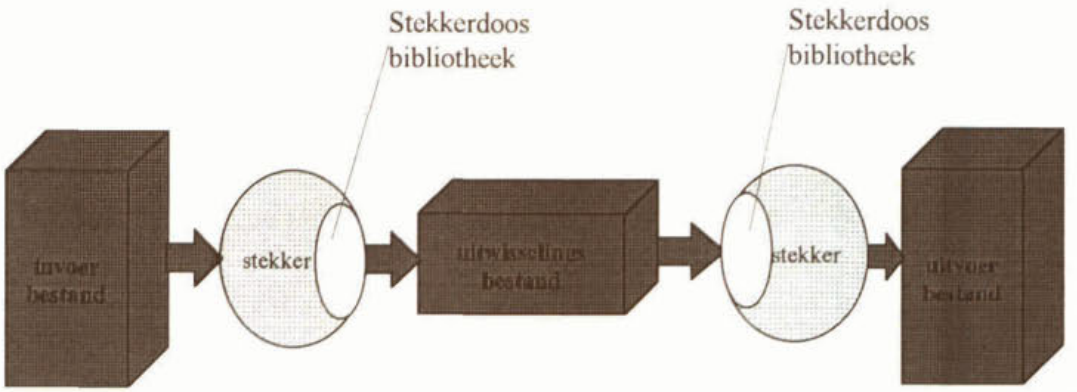


Figuur 1 Functie van de Stekkerdoos Water

Naast het gebruik van GW'96 is het ook mogelijk om afwijkingen en aanvullingen op GW'96 vast te leggen in een zogenaamd "bilateraal stuurbestand". Hiermee is het mogelijk om organisatie-specifieke classificaties uit te wisselen in aanvulling op de GW'96 classificaties.

1.3 Toepassingsgebied

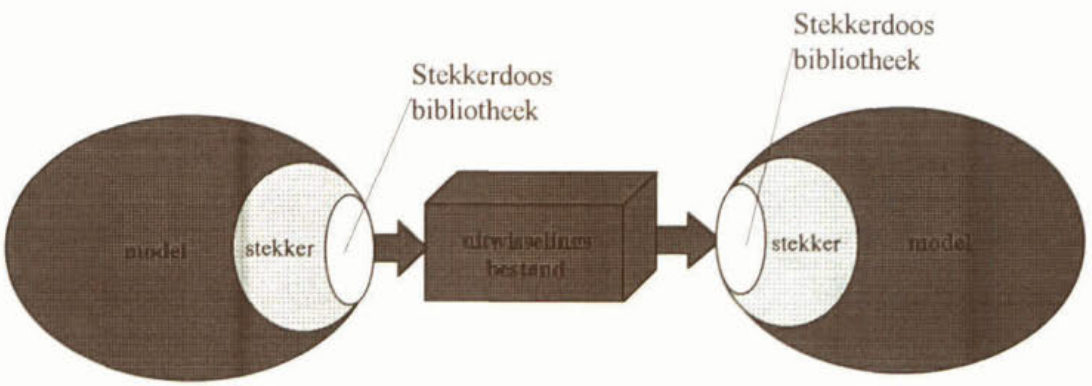
De Stekkerdoos Water is geschikt voor gebruik in programma's die gegevensbestanden omzetten naar uitwisselingsbestanden zoals aangegeven in .



FigF

Figuur 2: Omzetting van bestanden naar uitwisselingsbestanden

Tevens kan de Stekkerdoos Water worden geïntegreerd in mathematische modellen zoals waterbewegingsmodellen, grondwatermodellen en dergelijke zoals aangegeven in Figuur 3.



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 invoer bestanden naar de GW'96 classificatie (of de aanvullende bilateraal afgesproken classificatie). Deze vertaling (ook wel "**mapping**" genoemd) moet per model of per invoerbestand worden geprogrammeerd. De code waarin deze vertaling wordt verzorgd wordt de "**Stekker**" genoemd. Bij het gebruik van de Stekkerdoos Water binnen een model is de stekker dus geïntegreerd met het model.

Bij omzetting van databestanden naar uitwisselingsbestanden is de stekker een compleet 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 uit te wisselen, en de relaties tussen de waardereeksen en de relationele gegevens vast te leggen. Dit is in paragraaf 1.4.3 Uitwisseling van waardereeksen verder uitgewerkt.

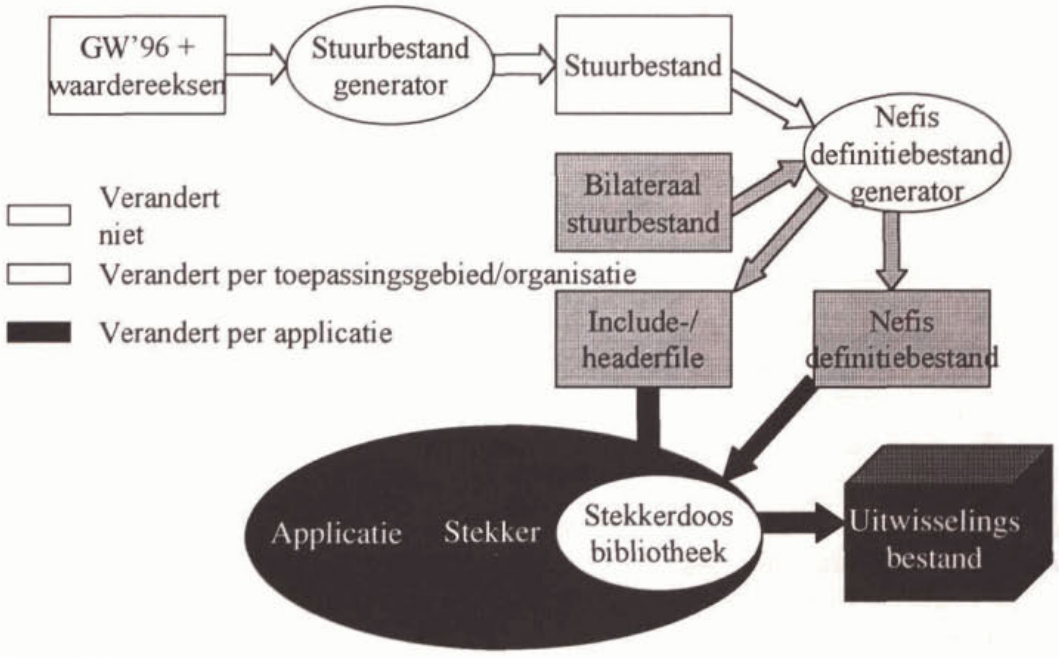
1.4 Functiebeschrijving

De Stekkerdoos Water bestaat uit een tweetal programma's en een bibliotheek/DLL met functies. Het eerste programma, de **Stuurbestandgenerator (CRESTBST)**, genereert vanuit de in SDW® opgeslagen **GW'96 classificatie** een zogenaamd **Stuurbestand**. Dit stuurbestand is een ASCII-bestand met de beschrijving van alle in de GW'96-classificatie voorkomende entiteiten en de bijbehorende gegevenselementen (attributen genoemd).

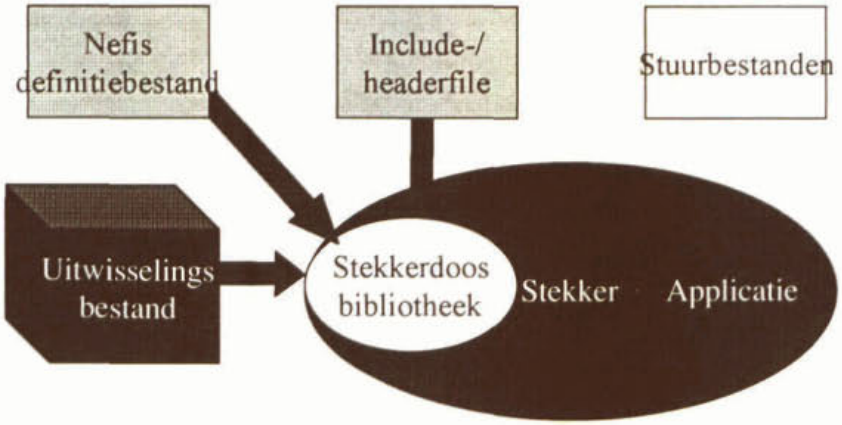
Vanuit dit stuurbestand wordt met behulp van het tweede programma, de **NEFIS-definitie-generator (CREDEF)**, een bestand aangemaakt waarin de structuur van het nog te vullen uitwisselingsbestand is vastgelegd (het zogenaamde **NEFIS-definitiebestand)**. Tevens genereert de NEFIS-definitie-generator bestanden die als zogenaamde "include-files" of "header-files" kunnen worden gebruikt in de te bouwen stekkers.

Het derde onderdeel van de Stekkerdoos Water is de DLL met functies, de zogenaamde **Stekkerdoos-software**. Deze functies kunnen vanuit een applicatieprogramma of een stekker worden aangeroepen voor het schrijven en lezen van de gegevens naar en van het uitwisselingsbestand.

Dit is grafisch weergegeven in Figuur 4: Systeemstructuur Exportstekker (voor de exportzijde en in Figuur 5 voor de importzijde)



Figuur 4: Systeemstructuur Exportstekker



Figuur 5: Systeemstructuur Importstekker

De stuurbestanden zijn nodig omdat daarin de structuur van GW'96 en het bilaterale bestand staan beschreven.

1.4.1 Uitwisseling van GW'96 geclassificeerde gegevens

GW'96 is een standaardbeschrijving van waterschapsgegevens volgens het relationele model. Dit houdt in dat al deze gegevens zijn beschreven in de vorm van tabellen. Deze tabellen worden **entiteittypen** genoemd. In Figuur 6 is een (vereenvoudigde) voorbeeldbeschrijving opgenomen voor entiteittype stuw. Per horizontale regel wordt één **object** (in dit geval één stuw) beschreven. In de kolommen worden de kenmerken (**attributen** genoemd) vastgelegd. Dit is grafisch weergegeven in Figuur 6.

Entiteittype: KST

Stuw identificatie	Soort stuw	Constructie hoogte	.....
stuw1	soort1	0.050	
stuw2	soort1	0.060	
stuw3	soort2	1.00	
stuw4	soort2	0.75	
.....	.....	....	

object 1
object 2
sleutel-attribuut
attribuut 2
attribuut 3

Figuur 6: Verkorte beschrijving van entiteittype KST (stuw)

De objecten binnen een entiteittype moeten altijd geadresseerd kunnen worden. Dit kan door de definitie van **sleutelattributen**. Sleutelattributen zijn die kolommen die samen voor elk object een unieke identificatie vormen. De waarden van de sleutelattributen worden **sleutelwaarden** genoemd en de namen van de attributen **sleutels.**

De Stuurbestandgenerator leest uit het SDW gegevensmodel de beschrijvingen van de entiteittypen met de bijbehorende attributen deze in het **GW'96-Stuurbestand**. Een "*" geeft aan dat het betreffende attribuut een sleutelattribuut is. (Alle GW'96 entiteittypen hebben minimaal één sleutelattribuut, en in de huidige versie van GW'96 maximaal 2. De Stekkerdoos staat echter meer sleutels toe)

In hoofdstuk 7 is een gedetailleerde beschrijving van het Stuurbestand met een voorbeeld opgenomen.

De Stekkerdoos-bibliotheek bevat functies voor het schrijven en lezen van één attribuutwaarde van één object per keer. Voor de adressering van het juiste object moeten de sleutelwaarden worden meegegeven aan de functie. Voor het wegschrijven van sleutelwaarden gelden speciale voorschriften. Zodra een veld wordt geschreven in een record welke nog niet bestaat, dan wordt dit record aangemaakt met als sleutelwaarden de opgegeven waarden. De sleutelwaarden kunnen op deze wijze nooit gewijzigd worden.

1.4.2 Uitwisseling aan de hand van bilaterale afspraken

Naast het gebruik van GW'96 is het ook mogelijk om aanvullende afspraken te maken. Dit is zinvol als gegevens moeten worden uitgewisseld die niet beschreven zijn in GW'96, of als de beschrijving van GW'96 niet voldoet. Deze afspraken kunnen worden vastgelegd in een aanvullend stuurbestand (**bilateraal stuurbestand** genoemd). Dit bestand heeft hetzelfde formaat (lay-out) als het GW'96 stuurbestand (zie hoofdstuk Bilaterale afspraken). Het bilaterale stuurbestand heeft prioriteit boven het GW'96 stuurbestand. Op deze wijze kunnen ook gewijzigde definities ten opzichte van GW'96 worden opgenomen. Het bilaterale stuurbestand wordt samen met het GW'96 stuurbestand omgezet naar een NEFIS-definitiebestand zoals ook blijkt uit Figuur 4: Systeemstructuur Exportstekker.

Nadat het NEFIS-definitiebestand is aangemaakt zijn de GW'96 structuren, aangepast/aangevuld volgens het bilaterale stuurbestand, beschikbaar voor de stekker. Binnen de Stekkerdoos Water wordt geen onderscheid meer gemaakt tussen bilateraal afgesproken definities en originele GW'96 definities. Wel bevat de Stekkerdoos-bibliotheek een functie waarmee de herkomst (GW'96 of bilateraal) van definities kan worden opgevraagd.

1.4.3 Uitwisseling van waardereeksen

De Stekkerdoos Water kan naast relationele gegevens (GW'96 en bilaterale afspraken) ook waardereeksen lezen en schrijven. Het verschil tussen waardereeksen en relationele gegevens is dat waardereeksen veelal lange reeksen van gegevens zijn met een vaste volgorde. Deze gegevens voldoen niet aan het relationele model omdat

- de volgorde van de waarden een betekenis heeft
- de waarden niet via sleutels benaderd kunnen worden en er geen relaties kunnen worden gelegd.

Het volgende voorbeeld geeft hiervan een illustratie. Figuur 7 geeft een voorbeeld van een relationele gegevensstructuur. Hierin zijn de stuwen te onderscheiden via de stuw-identificatie. In Figuur 8 is een voorbeeld gegeven van een waardereeks. Hierin wordt per "regel" een set van meetwaarden op één locatie op één tijdstip gegeven. De volgende regel bevat de meetwaarden voor het volgende tijdstip.

Relationeel model

stuw-identificatie	hoogte	breedte
stuw-1	40	60
stuw-2	50	80
...	....	....

adressering met behulp van een sleutel

Figuur 7: Voorbeeld relationele gegevensstructuur

Eén dimensionale waardereeks gemeten op een bepaalde locatie van bijvoorbeeld 0:00 tot 12:00 (equidistant met een interval van 10 minuten)

meetwaarde-1	meetwaarde-2
40	60
50	80
20	30
....	.....

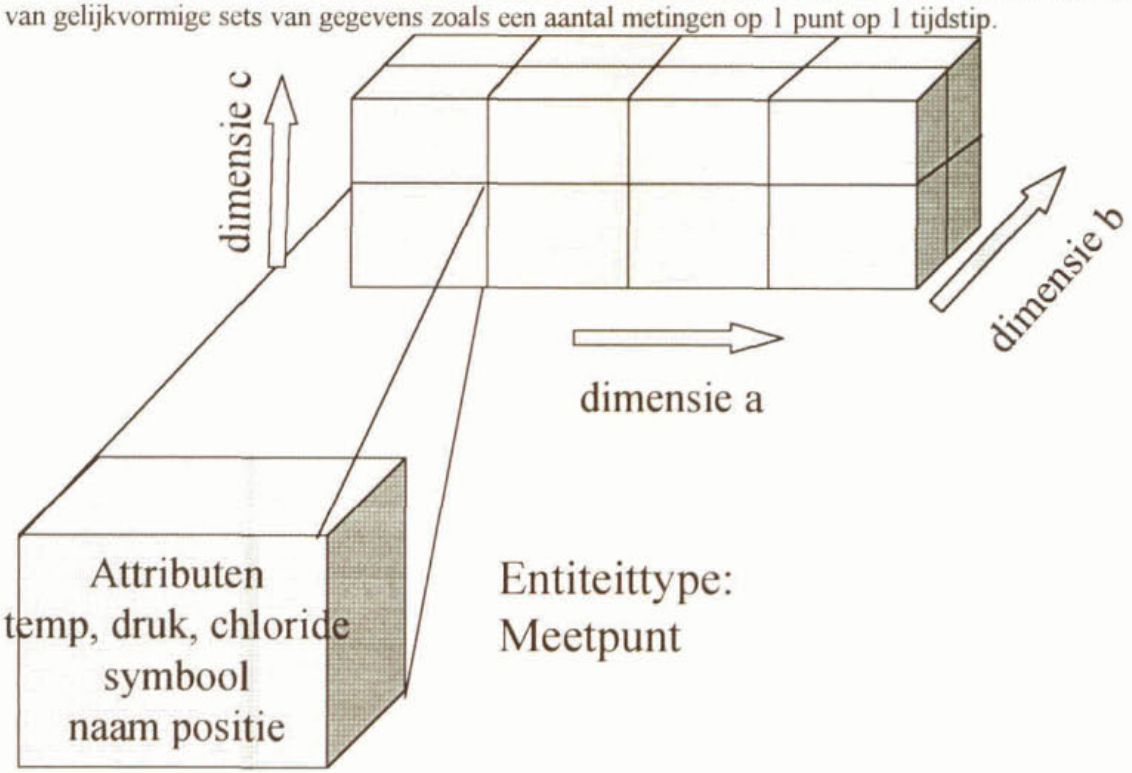
adressering met behulp van een volgnummer

Figuur 8: Voorbeeld van een waardereeks

Het is voor 1-dimensionale waardereeksen mogelijk om deze op te slaan in een relationeel model door aan alle metingen een sleutelwaarde (bijvoorbeeld het tijdstip) toe te voegen. Voor 2-dimensionale waardereeksen wordt dit al een stuk moeilijker en voor 3- of meer dimensionale waardereeksen moet veel administratie worden opgenomen om de 3-dimensionale structuur vast te leggen. Daarom biedt de Stekkerdoos Water speciale functies voor waardereeksen.

1.4.4 Waardereeksen in de Stekkerdoos Water

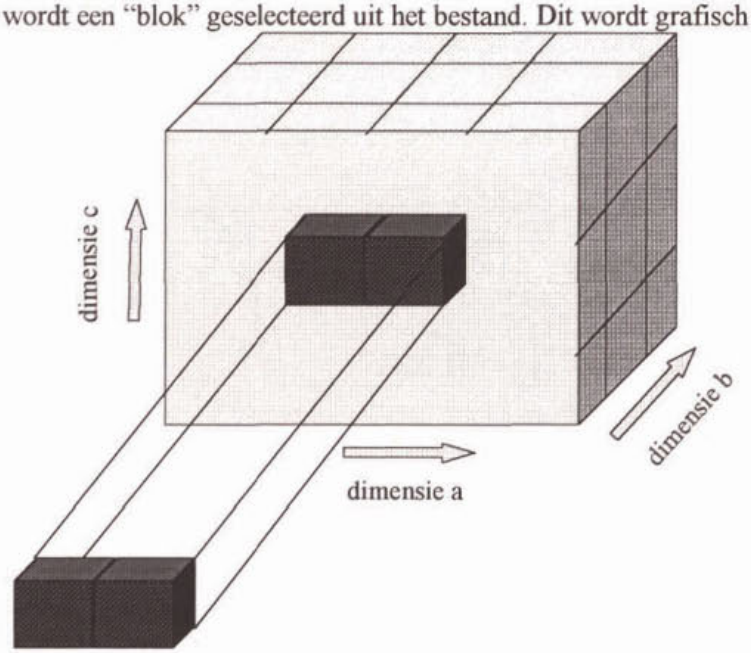
In de Stekkerdoos Water worden waardereeksen beschouwd als een meerdimensionale structuur van gelijkvormige sets van gegevens zoals een aantal metingen op 1 punt op 1 tijdstip.



Figuur 9: Structuur van waardereeksen

Figuur 9 is een grafische weergave van een 3-dimensionale waardereeks, waarbij per punt de waarden temperatuur, druk en chloride worden opgeslagen, samen met een symbool en een naam voor de locatie.

Bij het lezen of schrijven kan nu gerefereerd worden aan een positie(=locatie) door een waarde voor a, b en c op te geven. Door het opgeven van een start- en een eindpositie voor alle dimensies wordt een "blok" geselecteerd uit het bestand. Dit wordt grafisch weergegeven in Figuur 10.



Temp, druk, chloride op locatie (a2,b2,c2) en op locatie (a3,b2,c2)

Figuur 10: Het selecteren van gegevens uit een meerdimensionale waardereeks

De structuur van de waardereeks kan in een stuurbestand worden vastgelegd, door hiervoor een entiteittype te definiëren. Met uitzondering van één (de laatste), moeten voor de waardereeks-entiteittypen ook de dimensies en hun groottes worden opgegeven in het stuurbestand. De grootte van de laatste dimensie wordt aangegeven met een "*" teken.

In het voorbeeld van Figuur 9 zou in het stuurbestand een entiteittype waardereeks opgenomen kunnen worden met de volgende inhoud:

entiteittype: waardereeks; dimensies a=6, f=7, e=*
 attribuut: temperatuur
 attribuut: druk
 attribuut: chloride gehalte
 attribuut: symbool
 attribuut: naam geografische meetlocatie

In de stekker kan met behulp van de routine MAKWRD in "run-time" een waardereeks worden gecreëerd met een naam die in "run-time" wordt bepaald met gebruik van de definitie vanuit het stuurbestand. Op deze wijze is mogelijk om waardereeksen te creëren waarnaar wordt verwezen vanuit andere entiteittypen en namen te gebruiken die daarvan zijn afgeleid.

Metrische kenmerken-objectinformatie

identificatie	coördinaten stelsel	id puntenreeks	kleur	.....
stuw-1	RD	**stuw1rks**		
stuw-2	RD	**stuw2rks**		

puntenreeks: **stuw1rks**

interpolatie	x	Y	z

puntenreeks: **stuw2rks**

interpolatie	x	Y	z

Op het moment dat een waardereeks wordt aangemaakt wordt tegelijk de naam van de waardereeks geschreven in de entiteit/attribuut van waaruit naar de waardereeks wordt gerefereerd. De plaats van deze referentie wordt tevens opgeslagen bij de waardereeks, zodat deze plaats voor elke waardereeks achterhaald kan worden. De Stekkerdoos Water bevat geen "integriteitscontroles" met betrekking tot deze referenties. Dit houdt in dat als een verwijzing door een andere waarde wordt overschreven, dat de Stekkerdoos Water dit niet automatisch signaleert.

In tegenstelling tot de functies voor de relationele gegevens kunnen bij waardereeksen met één functieaanroep series van attribuutwaarden worden geschreven en gelezen, door een startpositie, een stapgrootte en het aantal stappen op te geven.

Opmerkingen

In de Stekkerdoos Water worden waardereeksen beschouwd als een meerdimensionale structuur van gelijkvormige sets van gegevens zoals bijvoorbeeld een aantal metingen op 1 punt op 1 tijdstip. De structuur van de sets moet in een stuurbestand worden vastgelegd, door hiervoor een entiteittype te definiëren. Met uitzondering van één (de laatste), moeten voor de waardereeks-entiteittypen ook de dimensies en hun groottes worden opgegeven in het stuurbestand. De grootte van die ene dimensie wordt aangegeven met een "*" teken.

Het zal veel voorkomen dat er meerdere voorkomens zijn van één waardereekstype. Hiervoor kan dan één en dezelfde waardereeksdefinitie worden gebruikt. De waardereeks zelf krijgt dan steeds een unieke naam.

Het is mogelijk om waardereeksen te koppelen aan een object, en omgekeerd kan vanuit een object de bijbehorende waardereeks worden gevonden.

2 Installatieprocedure

2.1 Systeemeisen

Voor een goede werking van de Stekkerdoos Water gelden de volgende systeemeisen:

- Voor het genereren van een GW'96 stuurbestand is nodig:
  - o een geïnstalleerde versie van GW'96
  - o SDW-Workstation met de module DataModelling (DM) versie 3.1* of 3.2* (SDW kan worden geleverd door CAP-Gemini)
  - o 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 NEFIS-definitiebestand is nodig:
  - o het GW'96-stuurbestand
  - o indien gewenst een bilateraal stuurbestand
  - o de NEFIS-definitie-generator

- Voor de stekkerbouw gelden de volgende eisen:
  - o een NEFIS-definitiebestand
  - o een 32-bits Windows computer (Windows NT of Windows 95)
  - o een 32-bits versie Fortran, C of C++ compiler (bij voorkeur Powerstation en/of C++ van Microsoft)

2.2 Platforms en portabiliteit

De Stekkerdoos Water is zoveel mogelijk platform onafhankelijk ontwikkeld, maar in eerste instantie gericht op de 32-bits Windows omgeving. Bij gebruik op andere platforms moeten zowel NEFIS als de Stekkerdoos Water worden geconverteerd. Voor NEFIS zijn bij het Waterloopkundig Laboratorium versies beschikbaar voor een uiteenlopende reeks van machines zoals HP-Unix, Sun-Solaris, Sillicon Graphics. Indien gewenst zal de Stekkerdoos Water zelf nog moeten worden geconverteerd naar de betreffende platforms.

2.3 Overzicht files op diskette

De distributiediskette bevat de volgende directories en bestanden:

2.3.1 Bibliotheek bestanden

De volgende bibliotheek bestanden staan op de directory \stkkrds\bin:

Stkkrds.dll	De DLL van de Stekkerdoos Water
Stkkrds_dll.lib:	Een aanvullende library voor gebruik van de DLL bij de Microsoft Fortran Powerstation compiler

2.3.2 Include/header files

De volgende include/header bestanden staan op de directory \stkkrds\bin:

Interface definities:

interf_dll.inc	Definitie van alle DLL routines t.b.v. Fortran
st_cintf.h	C-interface
st_intrf.inc	Alle benodigde include files voor gebruik bij de Microsoft Fortran Powerstation Fortran
rd1dwr.inc	Fortran interface voor rd1dwr
rd_key.inc	Fortran interface voor rd_key
rdcrnt.inc	Fortran interface voor rdcrnt
rdndwr.inc	Fortran interface voor rdndwr
wr_key.inc	Fortran interface voor wr_key
wr1dwr.inc	Fortran interface voor wr1dwr
wrcrnt.inc	Fortran interface voor wrcrnt
wrndwr.inc	Fortran interface voor wrndwr

Parameterfiles:

st_parms.h	C-parameters
st_parms.inc	Fortran parameters

NB: Aanpassing van deze parameters vereist tevens hercompilatie van de Stekkerdoos-bibliotheek en het programma CREDEF.

2.3.3 Datafiles

Error.dat	Het bestand met alle Stekkerdoos-foutmeldingen en deze staat op de directory \stkkrds\bin

2.3.4 Executables en SDW-scripts

De volgende executables en scripts staan op directory \stkkrds\bin:

Crestbst.scr:	Een SDW script voor het genereren van een stuurbestand vanuit een SDW versie van GW'96. (ASCII bestand)
Credef.exe:	Het programma CREDEF voor het omzetten van een stuurbestand naar een NEFIS-definitiebestand (Binair bestand)

2.3.5 Documentatie

Separaat is een gebruikershandeleiding beschikbaar
De volgende documentatie bestanden staan op de directory \stkkrds\doc:

Dll_desc.txt	De beschrijving van alle te gebruiken routines uit de DLL
Func_def.bas	De definitie van alle routines uit de DLL voor Visual Basic 5.0
Error.txt:	Een beknopte toelichting op Error.dat (ASCII bestand)

2.3.6 Stuurbestand GW96

Het stuurbestand van GW96 staat in de directory \stkkrds\bin

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

NB: De code 'jjmmdd' geeft de datum van het bestand aan.

2.3.7 C-interface

Voor het schrijven van C-programma's is een C-interface beschikbaar als file c_interf.c in de direcrtory stkkrds\c_code. In hoofdstuk 5 wordt verder op de DLL en dit C-interface ingegaan.

2.3.8 Voorbeelden

De voorbeelden uit de handleiding zijn beschikbaar als directories en files in de directory stkkrds\vrbld

2.4 Installatie-instructie

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

Er dient een directory \stkkrds te worden aangemaakt op de schijf waar de Stekkerdoos gebruikt gaat worden. De Stekkerdoos vereist dat de het bestand met de foutmeldingen (error.dat) staat in directory \stkkrds\bin. Wijziging van deze locatie vereist een hercompilatie van de bibliotheek.

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

2.5 Programmeur instructies

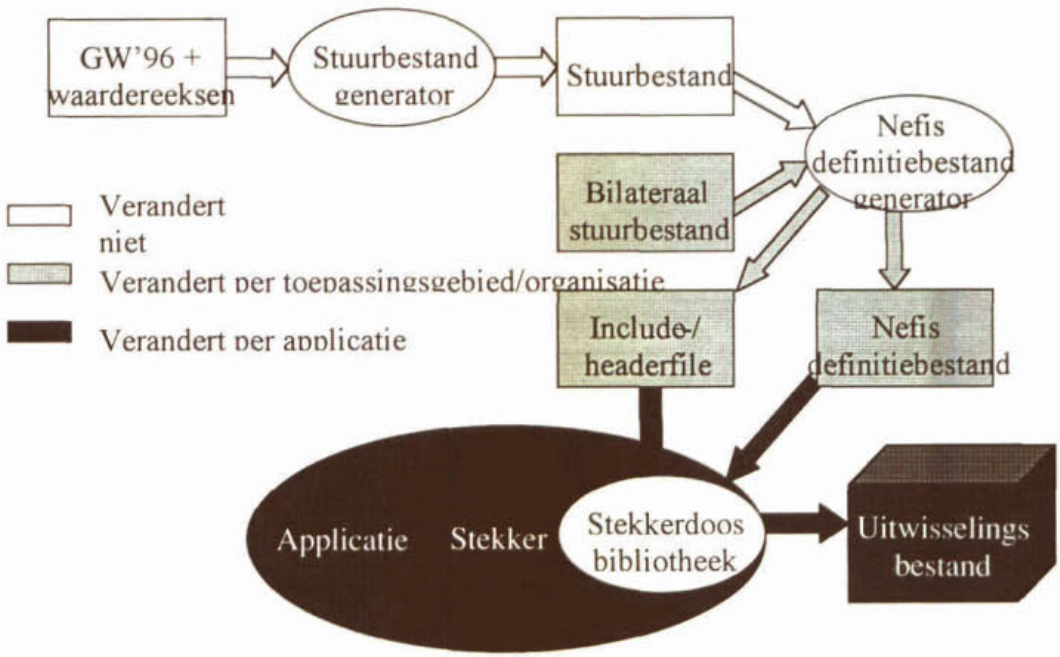
Er wordt van uit gegaan dat programmeurs die de Stekkerdoos Water gaan gebruiken voldoende op de hoogte zijn van de Fortran en/of C-compiler binnen hun organisatie. De programmeur kan de include of header files, zoals hierboven genoemd toepassen bij het bouwen van een applicatie. De beperking is dat deze specifiek voor de Microsoft compilers bestemd zijn

In hoofdstuk 5 wordt uitvoerig ingegaan op het gebruik van de DLL, voor Fortran en C, maar ook voor Visual Basic 5.0

Bij het ontwikkelen en testen van de stekkerdoos water is gebruik gemaakt van de Microsoft compilers Fortran Powerstation 4.0 en C/C++ versie 4.0. Beide zijn geinstalleerd met/onder de Microsoft Developer Studio. Hierbij zijn de diverse standaard 'options' en 'settings' gebruikt.

3 Systeembeschrijving

In Figuur 4 is een overzicht gegeven van het systeem. Voor het gemak wordt deze figuur hier nog een keer herhaald.



Figuur 11: Systeemoverzicht

In deze figuur zijn de volgende processen/bestanden te herkennen:

- GW'96 + waardereeksen
- Stuurbestandgenerator
- Stuurbestand
- Bilateraal stuurbestand
- NEFIS-definitiebestandgenerator
- NEFIS-definitiebestand
- Header-files
- Stekkerdoos-bibliotheek
- Applicatie (ook wel stekker genoemd)
- Uitwisselingsbestand

De kleur  geeft aan dat het dit onderdeel niet verandert bij gebruik in verschillende applicaties/stekkers.

De kleur  geeft aan dat het dit onderdeel alleen verandert bij gebruik in verschillende toepassingsgebieden. Aanbevolen wordt om per organisatie of per toepassingsgebied zo weinig mogelijk veranderingen aan te brengen in deze bestanden.

De kleur  geeft aan dat het dit onderdeel verandert per applicatie/stekker.

Deze processen en bestanden worden in de volgende hoofdstukken uitgewerkt.

3.1 GW'96 + waardereeksen

GW'96 (Gegevensstandaard Water) is een logisch gegevensmodel met daarin een classificatie van aan water gerelateerde gegevens opgesteld door de Unie van Waterschappen. Deze standaard is verkrijgbaar bij de Unie van Waterschappen. 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.

In GW'96 zijn een aantal mogelijkheden voor waardereeksen opgenomen. Deze zijn echter onvoldoende om geometrie en dergelijke te modelleren. In de toekomst zullen daarom ook nieuwe typen waardereeksen in GW'96 geclassificeerd worden. Zolang dit nog niet het geval is kunnen aanvullende en algemeen voorkomende waardereeksen in het stuurbestand worden opgenomen.

In het SDW model van GW'96 zijn entiteittypen geclusterd in superentiteittypen. Deze superentiteittypen hebben attributen die voor alle subentiteittypen van toepassing zijn. In de stuurtabel 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. Ook biedt de Stekkerdoos Water de mogelijkheid om te achterhalen welke attributen samen de primary key vormen. 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.

Aanbevolen wordt om in het stuurbestand dat met de Stekkerdoos Water wordt verspreid enkele aanvullende waardereeksen op te nemen.

3.2 Stuurbestandgenerator

Wanneer gebruiken

Gebruik dit programma alleen als vanuit SDW® een nieuw GW'96-stuurbestand moet worden gegenereerd. Zodra een stuurbestand beschikbaar is hoeft dit programma niet meer te worden gebruikt totdat een nieuwe versie van GW'96 beschikbaar komt. Indien geen SDW® beschikbaar is kan een stuurbestand van de Unie van Waterschappen worden betrokken. (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 [Ref 2].

Het programma heet Crestbst.scr. Voor de installatie moet Crestbst.scr bekend worden gemaakt aan SDW®. Daarna kan Stkkrds vanuit SDW® worden gedraaid. (Overigens geldt als randvoorwaarde dat GW'96 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 GW'96-classificatie

NB: De tijd die dit programma nodig heeft om te draaien kan afhankelijk van de snelheid van de computer en het netwerk oplopen tot ca 1 uur.

Eindsituatie

Na het draaien van de Stuurbestandgenerator is het GW'96-stuurbestand aanwezig dat als invoer kan dienen voor de NEFIS-definitie-generator.

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 Stuurbestand generator 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 GW'96 versienummer. Deze versie gegevens worden ook naar het uitwisselingsbestand geschreven. Deze gegevens zijn opvraagbaar met een bibliotheekfunctie.
4. Het versienummer van het bij het inpakken gebruikte classificatiestelsel zoals bepaald door de beheersorganisatie ervan, moet worden opgegeven.

3.3 Stuurbestand

Het stuurbestand is een bestand waarin eerst alle voorkomende attributen worden beschreven, en daarna alle entiteiten. In het stuurbestand zijn ook enige administratieve gegevens opgenomen zoals de datum waarop het stuurbestand is gegenereerd, de gebruikte versie van GW'96 en dergelijke. Een detailbeschrijving is opgenomen in paragraaf 6.3.

Opmerkingen

1. Het GW'96 stuurbestand kan met een willekeurige teksteditor waarmee ASCII-bestanden kunnen worden aangemaakt, worden gewijzigd.
2. De waardereeksen worden in de stuurbestanden gedefinieerd door achter een UvW-achtige code voor de betreffende entiteittype de dimensies aan te geven, waarbij altijd één dimensie (de laatste) wordt opgegeven met een sterretje "*"als indicatie dat de grootte van deze dimensie variabel is en pas "in run-time" wordt vastgesteld. Ook voor 1-dimensionale waardereeksen moet een "*" worden toegevoegd als teken dat het een gegevensblok betreft.

3.4 Bilateraal bestand

Het bilateraal bestand is van dezelfde vorm als het stuurbestand. In dit bestand kunnen gegevens van de GW'96 classificatie worden aangepast en aangevuld.

Als een entiteittype in het bilaterale bestand wordt geherdefinieerd dan geldt deze nieuwe definitie in plaats van de oude definitie.

Als een attribuut in het bilaterale stuurbestand wordt geherdefinieerd, dan geldt de nieuwe definitie automatisch voor alle entiteittypen waarin het betreffende attribuut wordt gebruikt.

Opmerkingen

1. Bij tegenstrijdigheden tussen de beide stuurtabellen krijgen de bilaterale afspraken de prioriteit boven de definities van GW'96.
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 de stuurbestanden worden gedefinieerd door achter de UvW code van de betreffende entiteittype de dimensies aan te geven, waarbij altijd één dimensie (de laatste) wordt opgegeven met een sterretje "*"als indicatie dat de grootte van deze dimensie variabel is en pas "in run-time" wordt vastgesteld.
4. Ook voor 1-dimensionale waardereeksen moet een "*" worden toegevoegd als teken dat het een waardereeks betreft.

3.5 NEFIS-definitie-generator (CREDEF)

Wanneer gebruiken

De NEFIS-definitie-generator (het programma CREDEF) moet worden ingezet als één van de beide stuurbestanden gewijzigd is. Als dit niet het geval is kan gebruik worden gemaakt van een reeds bestaand NEFIS-definitiebestand.

Uitgangspunten en randvoorwaarden

De NEFIS-definitie-generator vereist de aanwezigheid van een GW'96-stuurbestand. De aanwezigheid van een bilateraal bestand is optioneel.

Het draaien van de Nefis-definitie-generator (CREDEF)

De NEFIS-definitie-generator wordt gestart met het commando:

"CREDEF"

Het programma draait dan in een DOS-box onder Windows.
U wordt gevraagd de file namen op te geven:

- GW'96 stuurbestand
- bilateraal stuurbestand, indien gewenst
- de naam definitiebestand. De extensie hiervan moet zijn ".def"

NB: De tijd die dit programma nodig heeft om te draaien kan afhankelijk van de snelheid van de computer en het netwerk oplopen tot enkele minuten.

Het programma CREDEF genereert vanuit het stuurbestand de NEFIS-definitiefile.

Logica

Programma CREDEF vertaalt de stuurbestanden naar een NEFIS-definitiebestand. Onregelmatigheden die in de stuurbestanden worden geconstateerd worden gerapporteerd in een logfile.
De logica is als volgt:

- lees de administratieve gegevens van een eventueel aanwezig bilateraal bestand
- lees de attributen uit een eventueel aanwezig bilateraal bestand
- lees de administratieve gegevens uit het GW'96 stuurbestand
- lees de attributen vanuit het GW'96 stuurbestand

- lees de entiteiten vanuit het eventueel aanwezig bilateraal bestand
- lees de entiteiten vanuit het GW'96 stuurbestand

Eindsituatie

Na het draaien van dit programma zijn een NEFIS-definitiebestand, een C-headerfile en een Fortran-headerfile beschikbaar. Ook wordt een logfile credef.log gemaakt.

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. De NEFIS-structuur generator 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. Er worden een aantal extra gegevenselementen in het Nefis-definitiebestand gedefinieerd voor:
   - o de versie van de GW'96 classificatie,
   - o de naam van het SDW-model,
   - o de datum waarop het GW'96 stuurbestand is gegenereerd,
   - o de versie van het bilaterale stuurbestand,
   - o de toepassing (de reden waarom het bilaterale stuurbestand is aangemaakt)
   - o de datum waarop het bilaterale stuurbestand is aangemaakt,
   - o de identificatie code om na te gaan of het definitiebestand en het databestand bij elkaar horen.

   Deze gegevenselementen worden geheel binnen de Stekkerdoos-functies afgehandeld en zijn voor de stekkerbouwer afgeschermd. De betreffende informatie is met de opvraagfuncties te achterhalen.
4. Bij alfanumerieke velden in een stuurbestand worden eventueel opgegeven decimalen achter de punt genegeerd.
5. Datum velden zonder lengte worden verondersteld 8 posities lang te zijn.
6. Als in het stuurbestand een sleutelveld langer is dan 24 posities, dan wordt dit afgekapt op 24 posities.
7. Als geen type (alfanumeriek, numeriek of datum) is opgegeven in het stuurbestand dan wordt het veld verondersteld een alfanumeriek veld te zijn.
8. Als geen lengte is opgegeven in het stuurbestand dan wordt de lengte 8 verondersteld.
9. Entiteiten zonder sleutelattributen worden verondersteld waardereeksen te zijn. Hierbij moeten waarden zijn opgenomen voor de dimensies van het entiteittype.

3.6 NEFIS-definitiebestand

Het NEFIS-definitiebestand bevat definities van alle attributen en entiteiten uit het stuurbestand, in een zodanige vorm dat:

- alfanumerieke gegevens worden omgezet naar characterstrings van de lengte zoals opgenomen in het stuurbestand, naar boven afgerond naar veelvouden van 4.
- datumvelden worden opgenomen als characterstrings
- numerieke velden worden als volgt overgenomen:
  - o met decimale punt en totale lengte kleiner of gelijk 6 ⇒ Real*4
  - o met decimale punt en totale lengte kleiner of gelijk aan 15 ⇒ Real*8
  - o met decimale punt en totale lengte groter dan 15 ⇒ characterstring
  - o zonder decimale punt en een lengte kleiner of gelijk aan 9 ⇒ Integer*4
  - o zonder decimale punt en een lengte kleiner of gelijk aan 15 ⇒ Real*8
  - o zonder decimale punt en een lengte groter dan 15 ⇒ characterstring

In hoofdstuk 3.5 is beschreven hoe wordt omgegaan met fouten en ontbrekende informatie in de stuurbestanden.

In de 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.

De originele lengte zoals opgegeven in de stuurbestanden is opvraagbaar met de Stekkerdoosfunctie INQATT.

3.7 Header-files

Omdat de gebruiker van de Stekkerdoos-bibliotheek te ondersteunen worden vanuit de NEFIS-definitie-generator tegelijk bestanden aangemaakt waarin de Fortran- en C-declaraties van alle attributen zijn opgenomen. De stekkerbouwer hoeft de te gebruiken attributen zelf niet meer te declareren. Hierdoor worden vergissingen vermeden. Wel moet de stekkerbouwer de declaratie weten. Om deze te achterhalen kan het stuurbestand worden geraadpleegd, of ook de Stekkerdoosfunctie INQATT worden gebruikt.

Opmerkingen

1. De namen van de variabelen in de include-files zijn gelijk aan de namen van de UvW-codes van de betreffende gegevenselementen. In ANSI fortran mogen variabelen maximaal 6 posities lang zijn. Omdat de UvW-codes meer dan 6 characters lang zijn wordt aanbevolen om bij de bouw van de stekkers op dit punt van de ANSI standaard af te wijken. Indien de stekker strikt moet voldoen aan de ANSI standaard kunnen de standaard meegeleverde declaraties niet worden gebruikt

3.8 Stekkerdoos-bibliotheek

3.8.1 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 en deze code wordt aan het programme toegevoegd tijdens het laden (bij executie) ervan.

Bij het 'linken' om een 'executable' te maken moet ook informatie van deze DLL worden toegevoegd. Door nu gebruik te maken van meegeleverde 'include files' en een zeer specifieke library is dit proces foor Microsoft Fortran en C transparant voor de programmeur. Het is alsof bij 'gewone' (statische) libraries gebruikt bij het link proces.
De genoemde library is specifiek voor de Microsoft Fortran en C omgeving. Hij geeft de relatie aan tussen te gebruiken routine namen en de namen zoals ze in de DLL bekend zijn.

In deze handleiding blijven we de term bibliotheek gebruiken.
Specifieke informatie over de DLL wordt gegeven in hoofdstuk 5 DLL.

3.8.2 Performance

De Stekkerdoosfuncties kennen een reeks controles om de integriteit van de entiteiten te waarborgen. Het houdt o.a. in dat sleutelwaarden worden gecontroleerd, dat bij schrijven van een record via een sleutel alle attributen een initiële waarde krijgen.
Verder kan willekeurig een nieuw record van een entiteit worden toegevoegd en ook attributen van bestaande records worden toegevoegd en overschreven.

Gebruik van Stekkerdoos-functies om veel entiteiten of veel records van entiteiten naar de uitwisselingsfile te schrijven kan leiden tot performance problemen.
Om deze reden zijn er mogelijkheden aan de Stekkerdoos-functies toegevoegd. Hiermee kan de performance fors worden opgevoerd. Dit gaat echter ten koste van controles. Hiermee wordt de verantwoordelijkheid voor consistentie en integriteit van de entiteiten volledig bij de programmeur gelgegd.

De parameter tot betere performance is ingebouwd bij de routines WR-key en WRCRNT.
Teneinde de parameters van deze functies niet te wijzigen is gekozen om de parameter ERROR bij input een speciale betekenis toe te kennen. Zie verder hoofdstuk 5.

3.8.3 Functies

Onderstaand wordt een samenvatting van de Stekkerdoos-functies gegeven. Hoofdstuk 5 geeft een gedetailleerde beschrijving.

De functies zijn te verdelen in de volgende categorieën:

- openen en sluiten van een uitwisselingsbestand (OPNUWB en CLSUWB),
- creëren van entiteiten en waardereeksen in het databestand (MAKENT en MAKWRD),
- lezen en schrijven van attributen behorend bij entiteittypen (RD_KEY, WR_KEY, WRCRNT, RDCRNT en NXTOBJ),
- functies voor het lezen en schrijven van n-dimensionale waardereeksen. (WR1DWR, WRNDWR, RD1DWR en RDNDWR),
- opvraagfuncties het opvragen van:
  - bestandsinformatie (INQUWB),
  - alle in het uitwisselingsbestand aanwezige entiteiten en waardereeksen (INQCNT),
  - informatie van een entiteittype (INQENT) of een waardereeks (INQWRD) en
  - van informatie van een attribuut (INQATT).

De functies WRCRNT en RDCRNT (write- en read-current) zijn ontworpen om te vermijden dat als een reeks van attributen van één record binnen één entiteittype worden gevraagd of geschreven, dat te veel tijd verloren zou gaan met het zoeken en vergelijken van sleutelwaarden. In feite is hier dus sprake van een optimalisatie.

De functie NXTOBJ is ontworpen om de stekkerbouwer de mogelijkheid te bieden om sequentieel alle aanwezige records van één entiteittype te doorlopen zonder dat de sleutelwaarden bij hem bekend behoeven zijn.

Opmerkingen

1. De Stekkerdoos Water houdt een pointer bij naar het laatst benaderde (gelezen of geschreven record). Deze wordt bij het openen van een uitwisselingsbestand ingesteld op het eerste aanwezige object.
2. De Stekkerdoos Water is ongevoelig voor verschillen tussen hoofdletters en kleine letters in UvW-codes. De namen van de entiteittypen en attributen worden in hoofdletters in het

bestand opgenomen. Voor sleutelwaarden geldt dat bij het zoeken naar de juiste waarden geen onderscheid wordt gemaakt tussen hoofdletters en kleine letters.

3. Er zijn geen speciale voorzieningen getroffen voor het gebruik van de Stekkerdoos-functies in Fortran-90 applicaties/stekkers.
4. De include-/headerfiles met declaraties voor Fortran en C bevatten declaraties van alle gegevenselementen. Bovendien zijn een aantal parameters in deze files opgenomen, zoals het maximum aantal sleutels dat kan worden meegegeven en dergelijke.
5. Het gebruik van NEFIS routines buiten de Stekkerdoos-functies om kan de consistentie van het uitwisselingsbestand in gevaar brengen. Voor de juiste werking van de Stekkerdoos Water kan dan niet worden ingestaan. Daarom wordt het direct gebruik van NEFIS buiten de Stekkerdoos-functies om sterk afgeraden.
6. Alle door de Stekkerdoos Water gesignaleerde fouten en waarschuwingen worden naar een vaste logfile geschreven op het werkgebied, voorzien van een datum/tijd indicatie. Bovendien wordt het openen en sluiten van een uitwisselingsbestand naar de logfile geschreven. Een reeds bestaande logfile wordt aangevuld. Bovendien worden foutcodes doorgegeven aan de stekker. Aan de hand daarvan kan de stekkerbouwer zelf beslissen hoe een opgetreden fout wordt afgehandeld.
7. De GW'96 lees- en schrijfroutines kunnen niet worden gebruikt voor het lezen en schrijven van waardereeksen en omgekeerd kunnen de waardereeks-routines niet worden gebruikt voor het lezen van GW'96 attributen en entiteittypen. GW'96 gegevens worden geadresseerd via sleutelwaarden en waardereeks-informatie wordt geadresseerd via indices.
8. Gegevens die reeds in een uitwisselingsbestand zijn opgeslagen kunnen onbeperkt worden gewijzigd. (Dit geldt niet voor de sleutelvelden). Toevoegen is eveneens mogelijk. Verwijderen is niet mogelijk.
9. De te verwijderen velden kunnen als zodanig worden gemarkeerd met behulp van een was/wordt status. Hiervoor is een apart attribuut 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 de volgende codering: "N" nieuw, "D" gedeeltelijke herziening, "H" gehele herziening en "V" vervallen.

3.8.4 Overzicht van de functies

3.8.4.1 Openen en sluiten

OPNUWB: Open uitwisselingsbestand

CLSUWB: Sluit het data bestand

3.8.4.2 GW'96 uitwisseling

MAKENT: Creëer een entiteittype

WR_KEY : Schrijf een attribuutwaarde met sleutel

WRCRNT: Schrijf nog een attribuutwaarde naar het laatst geselecteerde record

RD_KEY: Lees een waarde van het eerste record dat overeenkomt met de opgegeven sleutelwaarden

RDCRNT : Lees nog een waarde van het laatst geselecteerde record

NXTOBJ: Selecteer het volgende record dat voldoet aan de sleutelwaarden

3.8.4.3 Uitwisseling van waardereeksen

MAKWRD : Creëer een waardereeks met de naam "waardereeksnaam" en structuur "entiteittype"

WR1DWR : Schrijf (een gedeelte van) een 1 dimensionale waardereeks

WRNDWR : Schrijf (een gedeelte van) een n dimensionale waardereeks (n <= 5)

RD1DWR : Lees (een gedeelte van) een 1 dimensionale waardereeks

RDNDWR : Lees (een gedeelte van) een n dimensionale waardereeks (n <= 5)

3.8.4.4 Informatie opvraagfuncties

INQUWB: Vraag bestandsinformatie

INQCNT: Vraag bestandsinhoud

INQENT: Vraag entiteittype informatie

INQWRD: Vraag waardereeks-informatie

INQATT: Vraag attribuut informatie

3.9 Applicatie/stekker

3.9.1 Algemene opmerkingen

1. Voor gewone entiteittypen geldt dat slechts één attribuut voor één object per keer kan worden benaderd.
2. Een blanco waarde is hetzelfde als spaties en wordt geïnterpreteerd als niet opgegeven.
3. Indien alle objecten van één entiteittype moeten worden gelezen of aangepast, wordt het gebruik van de functie NXTOBJ aanbevolen. Deze functie verschuift een interne pointer naar het volgende object dat voldoet aan de sleutelwaarden. Indien geen sleutelwaarden worden opgegeven wordt geschoven naar het eerstvolgende object. Hierdoor is het mogelijk om sequentieel alle aanwezige records van één entiteittype (met inachtneming van de opgegeven sleutelwaarden) te doorlopen zonder dat de sleutelwaarden op voorhand bekend behoeven zijn.
4. Indien meerdere malen hetzelfde object moet worden benaderd wordt het gebruik van de functies WRCRNT en RDCRNT sterk aanbevolen.
5. Het uitwisselingsbestand heeft de structuur zoals beschreven in het NEFIS-definitiebestand. Daarom moet bij uitwisseling altijd zowel het databestand als het definitiebestand worden geleverd alsmede een checksam bestand.
6. Op het moment dat een waardereeks wordt aangemaakt wordt tegelijk de naam van de waardereeks geschreven in de entiteit/attribuut van waaruit naar de waardereeks wordt gerefereerd. De plaats van deze referentie wordt tevens opgeslagen bij de waardereeks, zodat deze plaats voor elke waardereeks achterhaald kan worden. Indien geen naam voor een referentie entiteit wordt opgegeven dan wordt de waardereeks beschouwd als een losstaande waardereeks. De Stekkerdoos Water bevat geen "integriteitscontroles" met betrekking tot deze referenties. Dit houdt in dat als een verwijzing door een andere waarde wordt overschreven, dat de Stekkerdoos Water dit niet automatisch signaleert.

3.9.2 Het schrijven van een uitwisselingsbestand

De voorgeschreven volgorde van de functiegroepen is als volgt:
1. open het uitwisselingsbestand (OPNUWB)
2. maak 0, 1 of meer entiteittypen (MAKENT)
3. bepaal namen van 0, 1 of meer waardereeksen en creëer deze waardereeksen (MAKWRD)
4. schrijf 0, 1 of meer objecten naar de aangemaakte entiteittypen (WR_KEY en WRCRNT)
5. schrijf 0, 1 of meer waarden naar de waardereeksen (WR1DWR en WRNDWR)
6. herhaal de stappen 2 t/m 5 zoveel als nodig
7. sluit het uitwisselingsbestand

NB: Voordat gegevens van een object of een waardereeks kunnen worden weggeschreven moet de betreffende entiteittype of waardereeks zijn gecreëerd (met MAKENT en/of MAKWRD)

In een bijlage is een voorbeeld opgenomen van een stekker die een uitwisselingsbestand aanmaakt.

Opmerkingen

1. Waardereeksen kunnen maximaal 5-dimensionaal zijn.
2. Bij waardereeksen kan (hoeft niet) een referentie naar een andere entiteit worden opgegeven. Op deze wijze kan worden aangegeven waar de waardereeks bij hoort.
3. Het is niet toegestaan om een reeds ingevuld sleutelveld te wijzigen.

4. Niet gevulde velden worden geïnitialiseerd: Integers en reals op de maximaal mogelijke waarden en voor characterstrings wordt op de eerste positie een onleesbaar character (0-character) neergezet.
5. De stekker kan met behulp van de routine MAKWRD in "run-time" een waardereeks creëren met een naam die in "run-time" wordt bepaald. De maximale lengte van de naam van de waardereeks is 14 posities. Op deze wijze is mogelijk om waardereeksen te creëren waarnaar wordt verwezen vanuit andere entiteittypen.
6. Het is voor 1-dimensionale waardereeksen mogelijk om deze op te slaan in een relationeel model door aan alle metingen een sleutelwaarde toe te voegen. Voor 2-dimensionale waardereeksen wordt dit al een stuk moeilijker en voor 3- of meerdimensionale waardereeksen moet veel administratie worden opgenomen om de 3-dimensionale structuur vast te leggen.

3.9.3 Het lezen van een uitwisselingsbestand

De volgorde voor het lezen van een uitwisselingsbestand is
1. open het uitwisselingsbestand (OPNUWB)
2. lees van 0, 1 of meer opgeslagen objecten de gewenste attributen (RD_KEY en RDCRNT)
3. bepaal de namen van 0, 1 of meer waardereeksen en lees de gewenste gegevens van deze waardereeksen (RD1DWR en RDNDWR)
4. verwerk de gelezen gegevens
5. herhaal de stappen 2 t/m 4 zoveel als nodig
6. sluit het uitwisselingsbestand

Het is mogelijk om gericht te zoeken naar bepaalde informatie.
In een bijlage is een voorbeeldprogramma opgenomen welke alle informatie uit een uitwisselingsbestand leest en wegschrijft in een ASCII bestand.

Opmerkingen

De stekkerbouwer die de bestanden wil lezen hoeft niet te beschikken over de stuurbestanden, omdat alle informatie beschikbaar is, en opvraagbaar is vanuit het NEFIS-definitiebestand.

4 Ontwerp- en implementatiebeslissingen

1. Het include statement is geen ANSI Fortran. Toch wordt dit statement gebruikt om de onderhoudbaarheid van de code te vergroten. Op bijna alle platforms is wel een equivalente functie te vinden.
2. Momenteel zijn in GW'96 alle sleutelvelden alfanumeriek. Hier wordt in de Stekkerdoos Water mee gewerkt. Dit heeft als consequentie dat als in de toekomst met numerieke sleutelvelden wordt gewerkt deze eerst moeten worden geconverteerd naar alfanumeriek voordat deze aan de Stekkerdoos Water kunnen worden aangeboden.
3. De UvW-codes van de entiteittypen bevatten maximaal drie characters. De UvW-codes van attribuutnamen maximaal 8 characters. De Stekkerdoos Water staat voor namen van entiteittypen en waardereeksen een lengte van 14 posities toe.
4. In GW'96 komen geen 'Logical' velden voor. Daarom zijn in de Stekkerdoos Water geen faciliteiten voor 'logical' velden opgenomen.
5. Verder is een checksum aan de beide NEFIS-bestanden toegevoegd.
6. De NEFIS-definitiebestandgenerator neemt in het NEFIS-definitiebestand een unieke code op. Deze code wordt automatisch overgenomen in het uitwisselingsbestand. Als nu een uitwisselingsbestand wordt geopend met een definitiebestand dat een andere code bevat, dan wordt een waarschuwing gegeven als indicatie dat een ander NEFIS-definitiebestand wordt gebruikt dan het bestand waarmee de uitwisselingsfile is gegenereerd. Het is aan de stekkerbouwer om te beoordelen in hoeverre dit toelaatbaar is.
7. Door het gebruik van Nefis is het mogelijk om gegevens in een willekeurige volgorde naar het uitwisselingsbestand te schrijven en te lezen. Indien gebruik wordt gemaakt van extra performance, is de programmeur verantwoordelijk voor integriteit en consistentie en schrijven naar uitwisselingsbestand. Hierdoor is de Stekkerdoos Water bij uitstek geschikt om te worden ingezet in applicaties/stekkers die het uitwisselingsbestand direct benaderen (on-line).
8. Als de nefis-files gesloten zijn wordt van de beide nefis-bestanden (het definitiebestand en het data bestand) afzonderlijk de checksum berekend en weggeschreven in een apart bestand van het type ".chk". De berekening van de checksum gaat als volgt:
   Het bestand wordt opnieuw geopend daarna worden steeds per woord (16 bits) de modulo-3 waarden berekend. De resultaten per woord worden gesommeerd. Indien nu een stuk van het bestand mist dan levert de opsomming een te laag resultaat. Als een woord verminkt is dan is het resultaat net een beetje anders dan gewenst. De kans dat de ene storing de andere storing compenseert en daardoor onopgemerkt blijft is uitermate gering. Bij een foutief checksum resultaat wordt het data bestand "read-only" geopend.
9. Vanwege gebruik van Stekkerdoos-functies in Visual Basic en ook vanwege beperking van beheer- en onderhoudinspanning voor PC platform, is gekozen voor het leveren van een DLL. Ook andere dan Microsoft compilers kunnen hiermee omgaan.

5 DLL

5.1 Algemeen

Levering van een DLL op het PC platform heeft enkele voordelen. De belangrijkste is dat hiermee de Stekkerdoos-functies ook beschikbaar komen voor Visual Basic.
De andere is dat de huisige compilers van verschillende leveranciers voor het PC platform ook DLL's kennen. De DLL geeft een taal en compiler onafhankelijke toegang tot de Stekkerdoosfuncties. Het onderhouden van slechts één DLL voor de PC geeft forse kostenreducties bij beheer en onderhoud.

De Stekkerdoos-functies zijn in eerste instantie gemaakt voor toepassing binnen een omgeving voor de Microsoft compilers. Hiervoor zijn de benodigde include files, header files, specifieke library en C-code meegeleverd.

5.2 Microsoft Fortran

De library file "stkkrds-DLL.lib" is een statische library en bevat specifieke informatie om relaties tussen te gebruiken routinenamen en specifieke namen in de DLL vast te leggen. Deze library moet u opnemen in de lijst met libraries voor het 'linken' van het programma.

De file "Intelf-DLL.inc" bevat de declaraties van de Stekkerdoos-routines welke in de DLL zitten. Toepassen van deze 'include file' voorkomt tevens verkeerd gebruik van parameters van de diverse routines. U mag deze file niet wijzigen.

De functies voor lezen en schrijven van attributen en waardereeksen moeten overweg kunnen met real integer en character type van een attribuut of waardereeks. Binnen de routines van de Stekkerdoos is hiermee rekening gehouden.

Om de lees- en schrijfroutines voor ieder type attribuut en waardereeks identiek te houden, is gebruik gemaakt van de zogenaamde 'gemene' routines bij Microsoft Frotran. Hiervoor zijn de 'include files' Rd-key.inc, Rdcrnt.inc, Rd1dwr.inc, Rdndwr.inc en de groep Wr-key.inc, Wrcrnt.inc, Wr1dwr.inc en Wrndwr.inc gemaakt.

Deze twee groepen include files alsmede Interf-DLL.inc zijn samengebracht in de file St-intrf.inc.

Naast bovengenoemde include files kan de programmeur gebruik maken van Attdef..inc en St-parms.inc. De eerste is een declaratie van alle attribuut typen en is gegenereerd door progrmamma CREDEF.
De tweede geeft PARAMETER statements welke u eventueel kunt gebruiken ten behoeve van lengte array en character variabelen.

5.3 Microsoft C/C++

Ten behoeve van de C-programma's voor Microsoft C/C++ compiler zijn de volgende files beschikbaar :
De statische library Stkkrds.lib met de specifieke Fortran routine namen en de relatie met

DLL routine namen. Deze library moet u opnemen in de lijst met libraries voor het 'linken' van een executable.
Dan is er nog de file Stkkrds.h waarin de Fortran routines van de DLL als C-prototypes zijn gedeclareerd.
Het verband tussen deze Fortran routine namen en de DLL namen wordt dus gelegd via bovengenoemde routine.
De file St_Cintf.h geeft prototypen van de C-functies voor de Stekkerdoos.
De file C_interf.C geeft de C-code van de C-functies voor de Stekkerdoos.
U vindt hier de functie namen uit deze handleiding voorafgegaan met C_ en alles in hoofdletters.
U ziet ook dat binnen deze functies de Fortran DLL routines worden toegepast.
De Fortran routines voor de DLL handelen de verwerking van diverse typen (real, integer, character/string) correct af voor lezen en schrijven van attributen en waardereeksen.
Voor C is dit opgevangen door het type void pointer te gebruiken.

De plaats van genoemde files wordt gegeven in hoofdstuk 2.

5.4 Visual Basic

Aansluiten van de Visual Basic code op de DLL vereist kennis van de routine namen in de DLL en de parameters ervan.
Kennis hoe Microsoft Fortran parameters van routines behandeld en in de object code opneemt is gewenst en nodig.
Deze kennis is reeds toegepast om Visual Basic functie declaraties te maken van de Stekkerdoos routines. U vindt deze in de file Func_def.bas. Deze file is direct binnen Visual Basic beschikbaar.

De file geeft functie declaraties van alle Stekkerdoos routines in deze handleiding. Voor een aantal routines vindt u dezelfde namen, andere routines hebben nog een achtervoegsel van een letter. Dit zijn de routines die te maken hebben met lezen en schrijven van attributen en waardereeksen. Deze beide routines kunnen van verschillende typen zijn, zoals integer, real, character (de Fortran typen).
Visual Basic eist overeenkomst in typen parameters van functies. De routines in de DLL kennen hiervoor een parameter die van soort reference is. Verder wordt dat intern in de routines correct verwerkt.

Er zijn nog de volgende opmerkingen te maken :

- Character variabelen in Fortran (string) hebben 1 byte per character. In Visual Basic 5.0 hebben string variabelen 2 bytes. Hier moet dus omzetting plaatsvinden. De veilige manier is om een string om te zetten naar een byte array in VB;
- In Fortran liggen van character arrays alle bytes achter elkaar, bijvoorbeeld :
  CHARACTER sleutel(2) * 10 heeft 2 * 10 bytes achter elkaar. In VB moet u hier ook rekening mee houden; dus een array van 40 bytes en element 1 begint op positie 1 en element 2 op positie 21. De character lengte die ook moet worden gegeven is 20!;
- De gebruikte integers in Fortran zijn 4 bytes; dus type Long in VB;
- De parameters van de Fortran routines zijn van het soort Reference. De extra parameter voor lengte van strings die de compiler toevoegt, is van soort Value en type Long;
- In Fortran worden arrays aan routines doorgegeven by Reference en vanaf (eerste) array element. Denk hieraan bij Visual Basic, geef parameter door met array element en niet array naam. Bij de VB functie declaratie vindt u ook een parameter by Reference. Dit alles omdat een array in VB een struct is en extra bytes bevat;
- Bovenstaande informatie is verwerkt in de file Func_def.bas.

6 Detailbeschrijving bibliotheekfuncties

6.1 Inleiding

In deze paragraaf zijn de functiedefinities gegeven. Deze kunnen voor verschillende talen worden gebruikt, zie hiervoor hoofdstuk 5. De onderstaande beschrijvingen gebruiken Fortran definities. Van hieruit is vrij gemakkelijk te komen tot parameter types voor andere talen.

Bij iedere functie wordt per parameter vermeld welk type het is, of deze betrekking heeft op invoer of uitvoer (I/O), en is een omschrijving van de parameter opgenomen.

Opmerkingen van algemene aard:

- Indien een characterstring wordt opgevraagd vanuit het databestand moet de variabele waarin de waarde komt te staan eerst worden geïnitialiseerd. Indien dit wordt nagelaten dan kan het voorkomen dat het eerste deel van de string uit het bestand wordt gelezen, en het laatste deel nog een oude waarde bevat. (NEFIS leest slechts het aantal bytes dat in het bestand aanwezig is.)
- Voor namen van entiteiten en attributen worden 'lowercase' opgegeven namen ook 'lowercase' opgeslagen. Bij het benaderen van entiteiten en attributen wordt 'case-insensitive' gewerkt door altijd in 'uppercase' te vergelijken. Deze werkwijze wordt ook gehanteerd voor de waarden van sleutelvelden.
- Elke entiteit heeft een "object-wijzer". Deze wijst naar het laatst benaderde object. Deze objectwijzer wordt gebruikt door de functies RDCRNT en WRCRNT. Zolang er nog geen object benaderd is staat deze wijzer op 0 en kunnen de functies WRCRNT en RDCRNT nog niet worden gebruikt. De functie NXTOBJ verschuift de objectwijzer naar het eerstvolgende object dat voldoet aan de opgegeven sleutels. Als geen sleutels worden opgegeven wordt het eerstvolgende object geselecteerd. Als nog geen object is benaderd en er worden geen sleutels opgegeven, dan wordt het eerst voorkomende object geselecteerd.
- Als voor de functie RD_KEY geen sleutels worden opgegeven wordt het eerst voorkomende object geselecteerd. (Dit is immers het eerste object dat voldoet aan de (niet opgegeven) sleutels.)
- Bij het schrijven van een gewone entiteit worden alle attributen automatisch geinitialiseerd. Bij het schrijven van waardereeksen wordt niet automatisch geinitialiseerd.
  Zie ook performance in paragraaf 3.8.2. in verband met initialiseren.

6.2 Performance

Bij het schrijven van sleutels en attributen kan de performance opgevoerd worden ten koste van mindere controles. In paragraaf 3.8.2. wordt dit uitgelegd.
Als parameter is gebruikt ERROR om de performance te beinvloeden. Bij output krijgt deze de waarde ten behoeve van foutmeldingen.
Voor de routines WR_KEY en WRCRNT heeft hij de volgende betekenis als input parameter:

- **error = 0 (default)**

  kenmerk: **wel** controle op sleutelnamen, aantal sleutels en attribuutnamen;
  wel controle op voorkomen via gegeven sleutel.

  performance: de performance zal steeds verder afnemen bij het schrijven van en

steeds grotere aantal records (1000 en meer).

toepassing: uitermate geschikt voor het schrijven van losse gegevens waarbij consistentie een overwegende belangrijke rol speelt.

- **error = -1**

  kenmerk: **geen** controle op sleutelnamen, aantal sleutels en attribuutnamen; **geen** controle op voorkomen; per definitie betreft het een nieuw record.

  performance: er wordt een maximale performance verkregen; de verantwoordelijkheid wordt volledig bij de programmeur gelegd. Er is dus **geen** controle op het reeds bestaan van een record.

  toepassing: uitermate geschikt voor het schrijven van lange tijdreeksen (vele duizenden records) waarbij performance een belangrijke rol speelt.

  NB: in tegenstelling tot error = 0 en error = -1 wordt bij het schrijven van een nieuw record **niet** eerst alle attributen geinitialiseerd; alleen die attributen welke worden aangeboden worden naar het UWB geschreven. De niet aangeboden attributen zijn niet benaderbaar in het UWB.

- **error = -2**

  kenmerk: **wel** controle op sleutelnamen, aantal sleutels en attribuutnamen; **geen** controle op voorkomen; per definitie betreft het een nieuw record.

  performance: de verbetering van de performance wordt met name verkregen bij het schrijven van een steeds groter aantal records (meer dan 1000). Daar waar bij error = 0 de performance zal afnemen, blijft deze nu constant, onafhankelijk van het aantal te schrijven records. Dit is te verklaren doordat **niet** steeds opnieuw een steeds groter aantal records moet worden gecontroleerd op voorkomen.

  toepassing: uitermate geschikt voor het schrijven van lange tijdreeksen (vele duizenden records) waarbij performance een minder belangrijke rol speelt.

  NB: in tegenstelling tot error = 0 en error = -1 wordt bij het schrijven van een nieuw record **niet** eerst alle attributen geinitialiseerd; alleen die attributen welke worden aangeboden worden naar het UWB geschreven. De niet aangeboden attributen zijn niet benaderbaar in het UWB.

- **error = -1**

  kenmerk: **geen** controle op sleutelnamen, aantal sleutels en attribuutnamen; **wel** controle op voorkomen via gegeven sleutel.

  performance: de verbetering van de performance wordt met name verkregen bij het schrijven van een realteif klein aantal records (minder dan 1000).

Bij een groter aantal records is de performance vergelijkbaar met error = 0.

toepassing: uitermate geschikt voor het schrijven van kortere tijdreeksen waarbij performance en consistentie een belangrijke rol spelen.

6.3 Openen en sluiten van het uitwisselingsbestand

Bij de beschrijving van een aantal variabelen wordt een maximale lengte opgegeven. Veelal wordt hierbij de naam van een constante gebruikt. Deze constanten staan beschreven in 0: Beschrijving constanten

6.3.1 Open uitwisselingsbestand

Deze functie verzorgt het openen van het uitwisselingsbestand en het databestand, controleert of deze bestanden bij elkaar horen en controleert de integriteit middels een checksum.

OPNUWB (error, uitwisselingsbestand, status, omschrijving)

Parameters:	Type	I/O	Omschrijving
error	Integer	O	Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.
uitwisselingsbestand	Character*(*)	I	De naam van het te openen bestand met een eventuele pad-beschrijving (maximaal 200 characters). (De Stekkerdoos voegt zelf het gewenste bestandstype toe. Deze mogen dus niet worden gespecificeerd). Bestandstypen zijn .dat voor data file, .def voor definitie file, .chk voor checksum file na afloop.
status	Character*(*)	I	De status van het bestand: "NEW": open een nieuw bestand, een eventueel bestaand bestand wordt overschreven, "READ": open een bestaand bestand om te lezen en "UPDATE": wijzig een bestaand bestand
omschrijving	Character*(*)	I/O	Een tekstuele beschrijving voor het bestand, welke in dit bestand is/wordt opgenomen. Bij status = "NEW" wordt de opgegeven omschrijving naar het bestand geschreven (I) , bij status "READ" wordt de omschrijving gelezen (O). Bij status "UPDATE" wordt de omschrijving aangepast, behalve als een blanco string wordt opgegeven. In dit laatste geval blijft de omschrijving ongewijzigd en wordt teruggegeven via het veld omschrijving. (Maximale lengte = MXLEN1)

6.3.2 Sluit het uitwisselingsbestand

Met deze functie wordt het uitwisselingsbestand gesloten.

CLSUWB (error)

Parameters:	Type	I/O	Omschrijving
error	Integer	O	Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.

6.4 GW'96 uitwisseling

6.4.1 Creëer een entiteittype

Deze functie initialiseert een entiteittype en maakt het entiteittype bekend in het uitwisselingsbestand.

MAKENT (error, entiteittype)

Parameters:	Type	I/O	Omschrijving
error	Integer	O	Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.
entiteittype:	Character*(*)	I	Dit is een entiteittype uit de GW'96-classificatie, of uit de tabel met de bilaterale afspraken. (Maximale lengte is MXELEN)

6.4.2 Schrijf waarde met sleutel

Deze functie schrijft gegevens naar een attribuut behorend bij een record van een bepaald entiteittype naar het uitwisselingsbestand. Als het record nog niet bestaat dan worden tevens de sleutelwaarden weggeschreven en de overige waarden geïnitialiseerd.

WR_KEY (error, entiteittype, aantal_sleutels, sleutelnamen, sleutelwaarden, attribuutnaam, attribuutinhoud)

Parameters:	Type	I/O	Omschrijving
error	Integer	I/O	Waarde bij input zegt iets over performance, zie pararaaf 6.2. Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.
entiteittype:	Character*(*)	I	Dit is een entiteittype uit de GW'96-classificatie, of uit de tabel met de bilaterale afspraken (Maximale lengte = MXELEN)
aantal_sleutels	Integer	I	Het aantal opgegeven sleutels. Alle in het bestand aanwezige sleutels moeten worden gespecificeerd. (Maximaal MXNSLT)
sleutelnamen	Character(*)*(*)	I	De namen van de sleutels (Maximale lengte per sleutelnaam is MXSLEN)
sleutelwaarden	Character(*)*(*)	I	Een character array met de sleutelwaarden. De sleutelwaarden ZIJN verplicht. (Maximale lengte van sleutelwaarden is MXAVLN; aanbevolen wordt om deze lengte te beperken tot 24 characters.)
attribuutnaam	Character*(*)	I	De naam van het attribuut dat moet worden weggeschreven. Het is niet toegestaan om een reeds ingevuld sleutelveld te wijzigen. Voor schrijven van relatie-entiteittype (entiteittypen met alleen sleutelattributen) kan de attribuutnaam op blanco worden gezet als teken dat geen attribuutwaarde hoeft worden weggeschreven. (Maximale lengte is MXALEN)

attribuutinhoud	I	De inhoud van het attribuut. Het type moet overeen komen met het type zoals gedefinieerd in het stuurbestand. Hiervoor kan de declaratie worden gebruikt die in de include-/headerfile (attdef.inc of attdef.h) is opgenomen (de naam van deze variabele is gelijk aan de naam in het stuurbestand; veelal de UvW-code van het attribuut.) (Voor characterstrings geldt een maximale lengte van MXAVLN)

6.4.3 Schrijf een waarde naar het laatst geselecteerde record

Schrijf een attribuut in het laatst benaderde record van het opgegeven entiteittype. In de Stekkerdoos Water wordt per entiteittype bijgehouden welk record het laatst benaderd is. Deze functie kan worden gebruikt om te vermijden dat de Stekkerdoos Water steeds opnieuw gaat zoeken in het bestand naar het record met de juiste sleutelwaarden.

WRCRNT (error, entiteittype, attribuutnaam, attribuutinhoud)

Parameters:	Type	I/O	Omschrijving
error	Integer	I/O	Waarde bij input zegt iets over performance, zie pararaaf 6.2. Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.
entiteittype:	Character*(*)	I	Dit is een entiteittype uit de GW'96-classificatie, of uit de tabel met de bilaterale afspraken. (Maximale lengte is MXELEN)
attribuutnaam	Character*(*)	I	De naam van het attribuut dat moet worden weggeschreven. Deze waarde is verplicht. *en blanco is niet toegestaan.* (Maximale lengte is MXALEN)
attribuutinhoud		I	De inhoud van het attribuut. Het type moet overeen komen met het type zoals gedefinieerd in het stuurbestand. Hiervoor kan de declaratie worden gebruikt die in de *include-/headerfile is opgenomen* (de naam van deze variabele is gelijk aan de naam in het stuurbestand; veelal de UvW-code van het attribuut.) (Voor characterstrings geldt een maximale lengte van MXAVLN)

6.4.4 Lees een waarde van het eerst geselecteerde record

Met deze functie wordt eerst een record geselecteerd en direct daarna de betreffende waarde gelezen. Deze functie zoekt altijd vanaf het begin naar het eerste object dat voldoet aan de opgegeven sleutels.

RD_KEY (error, entiteittype, aantal_sleutels, sleutelnamen, sleutelwaarden, attribuutnaam, attribuutinhoud)

Parameters:	Type	I/O	Omschrijving
error	Integer	O	Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.
entiteittype:	Character*(*)	I	Dit is een entiteittype uit de GW'96-classificatie, of uit de tabel met de bilaterale afspraken.(Maximale lengte is MXELEN)
aantal_sleutels	Integer	I	Aantal sleutels. Dit aantal mag niet groter zijn dan het aantal in het bestand aanwezige sleutels. (Het aantal mag wel kleiner zijn.) In dat geval wordt het eerste record geselecteerd dat voldoet aan de opgegeven sleutelwaarden. (Maximaal aantal is MXNSLT) (Bij 0 sleutels wordt het eerste aanwezige object geselecteerd)
sleutelnamen	Character(*)*(*)	I	De namen van de sleutels (Maximale lengte per naam is MXSLEN)
sleutelwaarden	Character(*)*(*)	I	Een character array met de sleutelwaarden. (Maximale lengte per sleutelwaarde is MXAVLN, aanbevolen wordt om deze lengte te beperken tot 24 characters.)
attribuutnaam	Character*(*)	I	De naam van het attribuut dat moet worden gelezen. Deze waarde is verplicht en blanco is niet toegestaan. (Maximale lengte is MXALEN) (Deze variabele mag ook een sleutelnaam zijn.)

attribuutinhoud	O	De inhoud van het attribuut. Het type moet overeen komen met het type zoals gedefinieerd in het stuurbestand. Hiervoor kan de declaratie worden gebruikt die in de include-/headerfile is opgenomen (de naam van deze variabele is gelijk aan de naam in het stuurbestand; veelal de UvW-code van het attribuut.) (Voor een characterstring geldt een maximale lengte van MXAVLN) De variabele moet in de stekker worden geinitialiseerd.

6.4.5 Lees nog een waarde van het laatst geselecteerde record

Met deze functie kan herhaald van hetzelfde record worden gelezen zonder dat steeds aan de hand van sleutelwaarden gezocht hoeft te worden naar dit record.

RDCRNT (error, entiteittype, attribuutnaam, attribuutinhoud)

Parameters:	Type	I/O	Omschrijving
error	Integer	O	Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.
entiteittype:	Character*(*)	I	Dit is een entiteittype uit de GW'96-classificatie, of uit de tabel met de bilaterale afspraken. (Maximale lengte is MXELEN)
attribuutnaam	Character*(*)	I	De naam van het attribuut dat moet worden gelezen. Deze waarde is verplicht en blanco is niet toegestaan. (maximale lengte is MXALEN) (Dit mag ook de naam van een sleutel zijn.)
attribuutinhoud		O	De inhoud van het attribuut. Het type moet overeen komen met het type zoals gedefinieerd in het stuurbestand. Hiervoor kan de declaratie worden gebruikt die in de include-/headerfile is opgenomen (de naam van deze variabele is gelijk aan de naam in het stuurbestand; veelal de UvW-code van het attribuut.) (Voor een characterstring geldt een maximale lengte van MXAVLN) Deze variabele moet in de stekker worden geinitialiseerd.

6.4.6 Selecteer het eerstvolgende object

Met deze functie kan vanaf een bepaalde positie binnen een entiteittype het volgende object worden geselecteerd. Deze functie kan worden gebruikt als een volledig entiteittype moet worden uitgewisseld, zonder dat de sleutelwaarden relevant zijn. Deze functie zoekt vanaf de positie van de objectwijzer naar een object dat voldoet aan de opgegeven sleutels. (Objecten die nog voor de objectwijzer staan worden niet gevonden.) (Als het einde bereikt is dan resulteerd dit in foutnummer:50002)

NXTOBJ (error, entiteittype, aantal_sleutels, sleutelnamen, sleutelwaarden)

Parameters:	Type	I/O	Omschrijving
error	Integer	O	Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.
entiteittype:	Character*(*)	I	Dit is een entiteittype uit de GW'96-classificatie, of uit de tabel met de bilaterale afspraken. Maximale lengte is MXELEN)
aantal_sleutels	Integer	I	Aantal sleutels. Het aantal sleutels mag minder zijn dan het aantal in het bestand aanwezige sleutels. In dat geval wordt het eerste record geselecteerd dat voldoet aan de opgegeven sleutel waarden. (Maximaal aantal is MXNSLT) (Als geen sleutels worden opgegeven dan wordt het eerstvoorkomende object gekozen.)
sleutelnamen	Character(*)*(*)	I	De namen van de sleutels (Maximale lengte van de sleutelnamen is MXSLEN)
sleutelwaarden	Character(*)*(*)	I	Een character array met de sleutelwaarden. (Maximale lengte per sleutelwaarde is MXAVLN, aanbevolen wordt om deze lengte te beperken tot 24 characters.)

6.5 Uitwisseling van waardereeksen

6.5.1 Creëer een waardereeks

Deze functie creëert een waardereeks volgens een definitie welke is opgenomen in een van de stuurbestanden.

MAKWRD (error, entiteittype, waardereeksnaam, naam_ref_entiteit, naam_ref_attribuut, aantal_sleutels, sltlnamen_ref_object, sltwaarden_referentieobject)

Parameters:	Type	I/O	Omschrijving
error	Integer	O	Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.
entiteittype:	Character*(*)	I	Dit is een entiteittype uit de GW'96-classificatie, of uit de tabel met de bilaterale afspraken. (Maximale lengte is MXELEN)
waardereeksnaam	Character*(*)	I	De naam van de waardereeks. Deze naam kan vrij worden gekozen met een maximale lengte van MXELEN characters beginnend met een letter.
naam_ref_entiteit	Character*(*)	I	de naam van het entiteittype van waaruit wordt verwezen naar de waardereeks. Deze mag blanco zijn. In dat geval hoeft verder geen informatie van de referentie-entiteit te worden opgegeven. Indien wel een waarde wordt ingevuld, dan moeten ook alle sleutelwaarden worden opgegeven. (Maximale lengte MXELEN)
naam_ref_attribuut	Character*(*)	I	de naam van het attribuut van waaruit wordt verwezen naar de waardereeks. (maximale lengte MXALEN)
aantal_sleutels	Integer	I	Aantal sleutels (Maximaal MXNSLT)
sltnamen_ref_object	Character(*)*(*)	I	de namen van de sleutelattributen van het entiteittype van waaruit wordt verwezen naar de waardereeks (Maximale lengte is MXSLEN)

sltwaarden_ref_object Character(*)*(*)	I	de waarden van de sleutelattributen van het entiteittype van waaruit wordt verwezen naar de waardereeks (Maximale lengte van de waarden is MXAVLN; aanbevolen wordt een maximum van 24 posities aan te houden)

6.5.2 Schrijf een één dimensionale waardereeks

Met deze functie kan een waardereeks worden weggeschreven. Hierbij is het mogelijk om vanaf een bepaalde waarde te schrijven met een vaste interval.

WR1DWR (error, waardereeksnaam, attribuutnaam, start_index, aantal_stappen, stap_grootte, attribuutinhoud)

Parameters:	Type	I/O	Omschrijving
error	Integer	O	Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.
waardereeksnaam:	Character*(*)	I	Dit is de naam van de waardereeks waarheen moet worden geschreven. (Maximale lengte is MXELEN)
attribuutnaam	Character*(*)	I	De naam van het attribuut dat moet worden geschreven. Deze waarde is verplicht en blanco is niet toegestaan. (Maximale lengte is MXALEN)
start_index	Integer	I	Positie waar moet worden begonnen te schrijven
aantal_stappen	Integer	I	Het aantal waarden dat moet worden geschreven
stap_grootte	Integer:	I	De interval waarmee de gegevens moeten worden geschreven. 0 en 1 betekenen allebei stapgrootte 1
attribuutinhoud		I	Een array met attribuutwaarden voor de waardereeks. Het type moet overeen komen met het type zoals gedefinieerd in het stuurbestand. Hiervoor kan de declaratie worden gebruikt die in de include-/headerfile is opgenomen (de naam van deze variabele is gelijk aan de naam in het stuurbestand; veelal de UvW-code van het attribuut.)

6.5.3 Schrijf een n-dimensionale waardereeks

Met deze functie kan een meerdimensionale waardereeks worden weggeschreven. Per dimensie kan de startpositie, het interval en het aantal stappen worden opgegeven. Waardereeksen kunnen maximaal 5-dimensionaal zijn.

WRNDWR (error, waardereeksnaam, attribuutnaam, aantal_dimensies, start_indices_voor_de_dimensies, aantal_stappen_voor_de_dimensies, stap_grootte_voor_de_dimensies, attribuutinhoud)

Parameters:	Type	I/O	Omschrijving
error	Integer	O	Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.
waardereeksnaam:	Character*(*)	I	Dit is de naam van de waardereeks waarheen moet worden geschreven. (Maximale lengte is MXELEN.)
attribuutnaam	Character*(*)	I	De naam van het attribuut dat moet worden gelezen. Deze waarde is verplicht en blanco is niet toegestaan. (Maximale lengte is MXALEN)
aantal_dimensies	Integer	I	Aantal dimensies (Maximaal 5)
start_indices_voor_dims	Integer(*)	I	Een array met per dimensie de positie waar moet worden begonnen te schrijven
aantal_stappen	Integer(*)	I	Een array met het aantal waarden dat in de betreffende dimensie-richting moet worden geschreven
stap_grootte	Integer(*)	I	Een array met per dimensie een interval waarmee de gegevens moeten worden geschreven. 0 en 1 betekenen beide een stapgrootte van 1.
attribuutinhoud	(*)	I	Een array met attribuutwaarden waarin de schrijven moeten staan. Het type moet overeen komen met het type zoals gedefinieerd in het stuurbestand. Hiervoor kan de declaratie worden gebruikt die in de include-/headerfile is opgenomen (de naam van deze variabele is gelijk aan de naam in het stuurbestand; veelal de UvW-code van het attribuut.)

6.5.4 Lees een één dimensionale waardereeks

Deze functie is analoog aan de functie voor het schrijven van waardereeksen.

RD1DWR (error, waardereeksnaam, attribuutnaam, start_index, aantal_stappen, stap_grootte, buffer_grootte, attribuutinhoud)

Parameters:	Type	I/O	Omschrijving
error	Integer	O	Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.
waardereeksnaam:	Character*(*)	I	Dit is de naam van de waardereeks waarheen moet worden geschreven. (Maximale lengte is MXELEN)
attribuutnaam	Character*(*)	I	De naam van het attribuut dat moet worden geschreven. Deze waarde is verplicht en blanco is niet toegestaan. (Maximale lengte is MXALEN)
start_index	Integer	I	Positie waar moet worden begonnen te lezen
aantal_stappen	Integer	I	Het aantal waarden dat moet worden gelezen
stap_grootte	Integer:	I	De interval waarmee de gegevens moeten worden gelezen. o en 1 betekenen beide een stapgrootte van 1
buffer_grootte	Integer	I	De grootte van de buffer "attribuutinhoud" in bytes
attribuutinhoud		O	Een array met attribuutwaarden voor de waardereeks. Het type en de lengte moeten exact overeen komen met het type zoals gedefinieerd in het stuurbestand. Hiervoor kan de declaratie worden gebruikt die in de include-headerfile is opgenomen (de naam van deze variabele is gelijk aan de naam van het attribuut in het stuurbestand veelal de UvW-code van het attribuut.)

6.5.5 Lees een n-dimensionale waardereeks

Deze functie is analoog aan de functie voor het schrijven van waardereeksen.

RDNDWR (error, waardereeks, attribuutnaam, aantal_dimensies, start_indices_voor_de_dimensies, aantal_stappen_voor_de_dimensies, stap_grootte_voor_de_dimensies, buffer_grootte, attribuutinhoud)

Parameters:	Type	I/O	Omschrijving
error	Integer	O	Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.
waardereeksnaam:	Character*(*)	I	Dit is de naam van de waardereeks waarheen moet worden geschreven. (Maximale lengte is MXELEN)
attribuutnaam	Character*(*)	I	De naam van het attribuut dat moet worden gelezen. Deze waarde is verplicht en blanco is niet toegestaan. (Maximale lengte is MXALEN)
aantal_dimensies	Integer	I	Het aantal dimensies (Maximaal 5)
start_indices_voor_dims	Integer(*)	I	Een array met per dimensie de positie waar moet worden begonnen te lezen
aantal_stappen	Integer(*)	I	Een array met het aantal waarden dat in de betreffende dimensie-richting moet worden gelezen
stap_grootte	Integer(*)	I	Een array met per dimensie een interval waarmee de gegevens moeten worden gelezen. o en 1 betekenen beide een stapgrootte van 1.
buffer_grootte	Integer	I	Lente van de buffer "attribuutinhoud" in bytes.
attribuutinhoud		O	Een array met attribuutwaarden voor de waardereeks. Het type en de lengte moeten exact overeen komen met het type zoals gedefinieerd in het stuurbestand. Hiervoor kan de declaratie worden gebruikt die in de include-/headerfile is opgenomen. De naam van deze variabele is gelijk aan de naam in het stuurbestand. (veelal de UvW-code van het attribuut.)

6.6 Functies om informatie op te vragen

6.6.1 Vraag informatie van het uitwisselingsbestand

Met deze functie kunnen de administratieve kenmerken van het uitwisselingsbestand worden opgevraagd.

INQUWB (error, omschrijving_uwb, datum_laatste_wijziging_databestand, versie_gw_classificatie, naam_sdw_model, generatie_datum_stuurbestand, versie_bilaterale_afspraken, toepassing, datum_aanmaak_bil_stuurbestand)

Parameters:	Type	I/O	Omschrijving
error	Integer	O	Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.
omschrijving_uwb	Character*(MXLEN1)	O	De omschrijving opgenomen in het uitwisselingsbestand. (Maximale lengte is MXLEN1.)
datum_lst_wijziging_uwb	Character*(LENDAT)	O	De datum van de laatste wijziging in het databestand (wordt onder andere aangepast in de functie CLSUWB). Maximale lengte is LENDAT)
versie_gw_classificatie	Character*64	O	De versie van de GW'96-classificatie zoals vastgesteld door de beheersorganisatie van de GW'96-classificatie. (Maximale lengte is 64).
naam_sdw_model	Character*64	O	De naam van het SDW model waar het stuurbestand vanaf geleid is. (Maximale lengte is 64)
generatie_datum_stuurbst	Character*64	O	De datum waarop het stuurbestand is aangemaakt vanuit het SDW model (Maximale lengte is 64)
versie_bilaterale_afspr	Character*64	O	De versie van het bilaterale stuurbestand zoals opgenomen in het stuurbestand. (Maximale lengte is 64)
toepassing	Character*64	O	De toepassing waarvoor het bilaterale bestand is opgesteld. (Deze wordt overgenomen uit het bilaterale stuurbestand.) (Maximale lengte is 64)

datum_aanmaak_bil_stuurbst	Character*64	O	De datum van het bilaterale stuurbestand zoals opgenomen in het bilaterale stuurbestand. (Maximale lengte is 64)

6.6.2 Vraag bestandsinhoud

Met deze functie kunnen de namen van alle aanwezige entiteittypen en waardereeksen worden opgevraagd.

INQCNT (error, aantal_entiteittypen, entiteittypen, aantal_waardereeksen, waardereeksen)

Parameters:	Type	I/O	Omschrijving
error	Integer	O	Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.
aantal_entiteittypen	Integer	I/O	Bij Input het maximum aantal namen van entiteittypen dat in de array "entiteittypen" gelezen kan worden. (Maximaal MXNENT) Bij Output het aantal entiteittypen dat in het databestand is geschreven.
entiteittypen	character(*)*(*)	O	De namen van de entiteittypen (exclusief de waardereeks entiteittypen). (Maximale lengte per naam is MXELEN)
aantal_ waardereeksen	Integer	I/O	Bij Input het maximaal aantal namen van waardereeksen dat in de array "waardereeksnamen" gelezen kan worden. (Maximaal MXNRKS) Bij Output het aantal waardereeksen dat in het databestand is geschreven.
waardereeksnamen	character(*)*(*)	O	De namen van de waardereeksen in het databestand. (maximale lengte is MXALEN)

6.6.3 Vraag entiteittype informatie

Met deze functie kan informatie van entiteittypen worden opgevraagd. Met name het aantal records en de sleutelattributen zijn waardevolle gegevens. Ook kan met deze functie worden achterhaald of het entiteittype uit een bilateraal bestand komt of niet.

INQENT (error, entiteittype, aantal_records, aantal_attributen, attribuutnamen, aantal_sleutels, sleutelnamen, bilateraal_ja_nee)

Parameters:	Type	I/O	Omschrijving
error	Integer	O	Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.
entiteittype:	Character*(*)	I	Dit is een entiteittype uit de GW'96-classificatie, of uit de tabel met de bilaterale afspraken. (Maximale lengte is MXELEN)
aantal_records	Integer	O	Het aantal weggeschreven objecten
aantal_attributen	Integer	I/O	Bij Input het maximum aantal attribuutnamen dat in array "attribuutnamen" gelezen kan worden (exclusief de sleutels). Bij Output het aantal gedefinieerde attributen van het entiteittype.(Maximaal MXNATT)
attribuutnamen	Character(*)*(MXALEN))	O	De namen van de attributen van het entiteittype die geen sleutels zijn. (De lengte is van de namen is maximaal MXALEN).
aantal_sleutels	Integer	I/O	Bij Input het maximum aantal sleutelnamen dat in array "sleutelnamen" gelezen kan worden. Bij Output het aantal sleutels van het entiteittype. Maximaal MXNSLT)
sleutelnamen	Character(*)*(MXSLEN)	O	De namen van de sleutelattributen van het entiteittype.
bilateraal_ja_nee	Character*1	O	" J" => bilaterale definitie; "N" => definitie uit de GW'96-classificatie

6.6.4 Vraag waardereeksinformatie

Met deze functie kan informatie van waardereeksen worden opgevraagd. Met name de dimensies zijn waardevolle gegevens. Ook kan met deze functie worden achterhaald of de waardereeks een entiteittype gebruikt uit een bilateraal bestand of niet.

INQWRD (error, naam_waardereeks, gebruikte_entiteittype, aantal_dimensies, dimensies, aantal_attributen, attribuutnamen, bilateraal_ja_nee, naam_referentieentiteit, naam_referentieattribuut, aantal_sleutels_ref_obj, sleutelnamen_ref_obj, sleutelwaarden_ref_obj)

Parameters:	Type	I/O	Omschrijving
error	Integer	O	Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.
naam_waardereeks:	Character*(*)	I	Dit is de naam van de waardereeks.(maximale lengte is MXELEN)
entiteittype:	Character*(MXELEN)	O	Dit is een entiteittype uit de GW'96-classificatie, of uit de tabel met de bilaterale afspraken. (Maximale lengte is MXELEN)
aantal_dimensies	Integer	O	Het aantal in gebruik zijnde dimensies (maximaal 5, waarbij de laatste de vrij dimensie is)
dimensies	Integer(5)	O	Het bereik van de actuele dimensies (inclusief de vrije dimensie).
aantal_attributen	Integer	I/O	Bij Input het maximum aantal attribuutnamen dat gelezen kan worden in array "attribuutnamen". Bij Output het aantal attributen zoals opgenomen in de definitie van het entiteittype. (Maximaal MXNATT)
attribuutnamen	Character(*)*(MXALEN)	O	De namen van alle attributen van het entiteittype. (Maximale lengte per attribuutnaam is MXALEN)
bilateraal_ja_nee	Character*1	O	" J" => bilaterale definitie; "N" => definitie uit de GW'96-classificatie
naam_referentieentiteit	Character*(MXELEN)	O	de naam van het entiteittype van waaruit wordt verwezen naar de waardereeks (Maximale lengte is MXELEN)

naam_referentieattribuut	Character*(MXALEN))	O	de naam van het attribuut van waaruit wordt verwezen naar de waardereeks (Maximale lengte is MXALEN)
aantal_sleutels_ref_obj	Integer	I/O	Bij Input het maximaal aantal sleutels dat gelezen kan worden in array "sleutelnamen_referentieobject" en "sleutelwaarden_referentieobject". Bij Output het aantal gelezen sleutelwaarden.(Maximaal MXNSLT)
sleutelnamen_ref_obj	Character(*)*(MXSLEN)	O	de namen van de sleutelattributen van het entiteittype van waaruit wordt verwezen naar de waardereeks
sleutelwaarden_referentieobject	Character(*)*(MXAVLN)	O	de waarden van de sleutelattributen van het entiteittype van waaruit wordt verwezen naar de waardereeks. (Maximale lengte is MXAVLN)

6.6.5 Vraag attribuut informatie

Met deze functie kan informatie van attributen worden opgevraagd. Voor deze functie is met name de attribuutlengte van belang, omdat dit weergeeft hoe het gegeven in GW'96 is gedefinieerd.

INQATT (error, attribuutnaam, attribuuttype, attribuutlengte, bilateraal_ja_nee, NEFIS_type, NEFIS-lengte)

Parameters:	Type	I/O	Omschrijving
error	Integer	O	Een rapportage van het resultaat van de functie (error > 0 ==> waarschuwing, error < 0 ==> fatale fout.
attribuutnaam	Character*(*)	I	De naam van het attribuut waar gegevens van moeten worden opgevraagd. (Maximale lengte is MXALEN)
attribuuttype	Character*16	O	Het type van het attribuut zoals gedefinieerd in de GW'96-classificatie of in het bilaterale bestand.
attribuutlengte	Character*16	O	De lengte van het attribuut zoals gedefinieerd in de GW'96-classificatie of in het bilaterale bestand
bilateraal_ja_nee	Character*1	O	" J" => bilaterale definitie; "N" => definitie uit de GW'96-classificatie
NEFIS_type	Character*8	O	Het type van het attribuut zoals deze in NEFIS wordt opgeslagen.
NEFIS-lengte	Integer	O	De lengte van het attribuut in characters of bytes zoals opgeslagen in NEFIS.

6.7 Beschrijving constanten

Deze constanten zijn opgeslagen in de include-file ST_PARMS.INC. Deze worden ook intern in de Stekkerdoos Water gebruikt en mogen daarom niet worden aangepast.

MXLEN1 = 100	Maximale lengte omschrijving bestanden
MXNSLT = 10	Maximaal aantal sleutels
MXNATT = 50	Maximaal aantal attributen (geen sleutels)
MXAVLN = 400	Maximale lengte attribuut- of sleutelwaarde in bytes
MXSLEN = 14	Maximale lengte sleutelnaam (in stuurbestand)
MXALEN = 14	Maximale lengte attribuutnaam (in stuurbestand
MXELEN = 14	Maximale lengte entiteitnaam (in stuurbestand)
LENDAT = 24	Lengte datum-tijd string
MXNENT = 500	Maximaal aantal entiteiten per bestand
MXNRKS = 500	Maximaal aantal waardereeksen per bestand

Voor C-programma's gelden de volgende waarden: beschreven in (ST_PARMS.H)

MXLEN1 =	101	
MXNSLT =	11	
MXNATT =	51	
MXAVLN =	401	
MXALEN =	15	
MXSLEN =	15	
MXELEN =	15	
MXNENT =	500	Maximaal aantal entiteiten per bestand
MXNRKS =	500	Maximaal aantal waardereeksen per bestand
LENDAT =	25	Lengte datum-tijd string

7 Detailbeschrijving stuurbestanden

Dit hoofdstuk beschrijft het formaat van de stuurbestanden.

Beide stuurbestanden zijn als volgt opgebouwd:
1. Het eerste deel bevat enige administratieve informatie:
2. Het tweede deel bevat een beschrijving van alle gegevenselementen
3. 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).

7.1 GW-classificatie

7.1.1 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

7.1.2 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 in het NEFIS-definitiebestand.

Volgende regel:

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

7.1.3 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 15: de UvW-code van het entiteittype of de naam van de waardereeks
3. kolom 16: spaties
4. kolom 17 t/m 41: als het een waardereeks betreft, dan opsomming van de dimensies, gescheiden door een ",". De vrije dimensie wordt aangegeven door een "*" teken. Voor elke waardereeks moet altijd als laatste één vrije dimensie worden opgegeven.
5. kolom 42: spatie
6. kolom 43 t/m 132: de naam van het entiteittype zoals opgenomen in GW'96 of de naam van de waardereeks. Deze 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 welke geen waardereeks is 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.

7.2 Bilaterale afspraken

7.2.1 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.

7.2.2 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

7.2.3 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)
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: Dit veld wordt niet meegenomen in het NEFIS-definitiebestand.

Volgende regel:

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

7.2.4 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. 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 15:	Een unieke code van het entiteittype (de zogenaamde bilaterale code)
3.	kolom 16:	spatie
4.	kolom 17 t/m 41:	als het een waardereeks betreft, dan opsomming van de dimensies, gescheiden door een ",". De vrije dimensie wordt aangegeven door een "*" teken. Voor elke waardereeks moet altijd één vrije dimensie (de laatste) worden opgegeven.
5.	kolom 42:	spatie
6.	kolom 43 t/m 132:	een willekeurige naam van het entiteittype Deze 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:	een 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 welke geen waarde-reeks 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 de uitwisselingsfile of definitiefile.

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

7.3 Voorbeeld 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 rioolwaterzuiveringsinstallatieAlfanumeriek 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 Overzicht foutmeldingen

De Stekkerdoos genereert foutmeldingen en waarschuwingen. De meldingen worden geschreven naar een logfile. Deze logfile wordt steeds aangevuld. Het voordeel hiervan is dat oude meldingen bewaard blijven. Het nadeel hiervan is dat dit bestand steeds groter wordt en af en toe handmatig moet worden verwijderd.

De foutmeldingen zijn onderverdeeld in de volgende categorieën:

- Foutmeldingen CREDEF
- Waarschuwingen Stekkerdoos Water
- Foutmeldingen Stekkerdoos Water
- NEFIS foutmeldingen

Per fout is aangegeven of deze fout een gebruikersfout is of een fout die moet worden doorgegeven aan de beheersorganisatie: B → Fout die moet worden doorgegeven aan de ontwikkelaars, G → een fout die door de gebruiker is veroorszaakt (door bijvoorbeeld afwezigheid van gegevensbestanden). G/B betetekt dat de fout mogelijk veroorzaakt is door de gebruiker.

8.1 Foutmeldingen CREDEF

00001 Filenaam %1 is langer dan 12 karakters
G → Vervallen

00002 I/O error bij openen filenaam %1
G → Fortranfout in routine OPNFLS bij opvragen van de status of bij het openen van het definitiebebestand, stuurbestand of het bilateraal bestand. Controleer de schijf en de aanwezige files.

00003 I/O error bij sluiten van file %1
B → Sluiten van het GW96-, of bilateraalstuurbestand of NEFIS-definitiebestand is mislukt.

00004 Attribuut %1 heeft een onbekend attribuuttype %2 in stuurfile
G → Het attribuuttype moet zijn: "REAL", "INTEGER", "CHARACTER". Het onbekende type wordt omgezet naar "CHARACTER"

00005 I/O error tijdens lezen attributen uit stuurfile
G/B → Controleer het GW96 en bilateraal stuurbestand.

00006 I/O error %1 tijdens schrijven attributen in definitiefile
G/B → Controleer de schijf en de files.

00007 I/O error %1 tijdens lezen attributen uit definitiefile
G/B → Controleer het definitiebestand en de schijf.

00008 Error bij opvragen systeemtijd
B → Fortran fout in routine ADDINF

00009 Error tijdens lezen administratie uit stuurfile
G → Controleer de administratiegedeeltes van de beide stuurbestanden

00010 Te veel administratieregels in stuurfile
G → Verwijder de overtollige regel uit het betreffende stuurbestand

00011 I/O error tijdens schrijven administratie in definitiefile
B → Controleer de schijf en de files.

00012 I/O error tijdens lezen van entiteit %1 uit de stuurfile
G/B → Controleer de beide stuurbestanden

00013 I/O error %1 tijdens schrijven van entiteit %2 in de definitiefile
B → Fortran fout in routine DEFENT.

00014 Te veel attributen in de stuurfile voor entiteit %1
G → Er is een maximum gesteld aan het aantal attributen per entiteit. Indien dit aantal niet voldoende blijkt moet de bibliotheek opnieuw worden aangemaakt met een hogere parameter. Dit is een taak van de beheersorganisatie van de stekkertdoos.

00015 Definitiefile %1 bestaat al
G → Verwijder (of verplaats) het bestaande definitiebestand en probeer opnieuw.

00016 File %1 bestaat niet
G → Stuurbestand of bilateraal bestand bestaat niet. Dit stuurbestand kan worden gemaakt met het SDW programma CRESTBST. Het bilaterale bestand met een teksteditor.

00017 Attribuut %1 is reeds gedefinieerd (waarschijnlijk in het bilaterale bestand)
G → Attributen mogen maar 1 maal worden gedefinieerd. Het kan zijn dat hetzelfde attribuut herhaald voorkomt binnen 1 stuurbestand. Het kan echter ook voorkomen een bepaald attribuut in GW96 en ook in het bilaterale stuurbestand wordt gedefinieerd. In dat geval moet dit bericht worden beschouwd als een waarschuwing. De definitie in het GW96 stuurbestand wordt dan verder genegeerd.

00018 Entiteit %1 heeft geen attributen
G → Elke entiteit moet minimaal 1 attribuut hebben. De entiteiten zonder attributen moeten uit het stuurbestand worden verwijderd.

00019 Entiteit %1 is reeds gedefinieerd (waarschijnlijk in het bilaterale bestand)
G → Entiteiten mogen maar 1 maal worden gedefinieerd. Het kan zijn dat dezelfde entiteit herhaald voorkomt binnen 1 stuurbestand. Het kan echter ook voorkomen dat een bepaalde entiteit in GW96 en ook in het bilaterale stuurbestand wordt gedefinieerd. In dat geval moet dit bericht worden beschouwd als een waarschuwing. De definitie in het GW96 stuurbestand wordt dan verder genegeerd.

00020 Attribuut %1 zoals opgegeven voor entiteit %2 bestaat niet
G → Controleer de beide stuurbestanden

00021 Entiteit %1 is een waardereeks, maar heeft ook sleutelattributen
G → In waardereeksen mogen geen sleutels voorkomen. Maak daarom een keuze of de betreffende entiteit een waardereeks is of niet. Zo ja verwijder dan de "*" achter de attributen van deze entiteit.

00022 Errorfile %1 niet aanwezig
B → Vervallen

00023 Logfile is niet correct geopend
B → Vervallen

00024 Entiteit %1 is geen waardereeks en heeft geen sleutelattribuut
G → Gewone entiteiten moeten minimaal 1 sleutelattribuut hebben. Voeg een "*" toe aan het een attribuut van de betreffende entiteit, of een dimensie aan deze entiteit.

00025 Attribuut %1 heeft geen lengte; defaultlengte 8
G → De lengte wordt automatisch op 8 gezet.

00026 Lengte van array om %1 op te slaan, is onvoldoende
G → Geef een grotere buffer op voor de namen van de waardereeksen en/of entiteiten en/of attributen en/of sleutelnamen

00027 Attribuut %1 komt meerdere keren voor in entiteit %2
G → Verwijder in het stuurbestand het betreffende attribuut uit de entiteit.

00028 Sleutelattribuut %1 van entiteit %2 is niet van het type CHARACTER
G → Pas het betreffende stuurbestand aan

00029 Attribuut %1 gedefinieerd in NEFIS definitiebestand
G → Dit is een melding van het correct verlopen van het aanmaken van een attribuut.

00030 Entiteit %1 gedefinieerd in NEFIS definitiebestand
G → Dit is een melding van het correct verlopen van het aanmaken van een entiteit.

8.2 Waarschuwingen Stekkerdoos Water

50000 File %1 is platform afhankelijk
B → Het uitwisselingsbestand moet platform onafhankelijk zijn zodat dit getransporteerd kan worden naar een ander platform. De Stekkerdoos Water genereert automatisch platform onafhankelijke bestanden.

50001 Identificaties definitie- en databestand komen niet overeen
G → Dit is een controle of het '.dat' bestand gegenereerd is met het bijgevoegde '.def' bestand. Indien de gebruiker zeker weet dat alle nodige definities correct in het '.def' bestand staan, dan kan deze melding worden genegeerd. Zo niet dan is de gegevensuitwisseling onbetrouwbaar.

50002 In entiteit %1 is geen record gevonden dat voldoet aan de opgegeven sleutels
G → Controleer de sleutelwaarden

50003 Entiteit %1 is niet aanwezig
G → Controleer de naam van de entiteit (vergelijk met het stuurbestand)

50004 Entiteit %1 is al aanwezig in uitwisselingsbestand
G → Er wordt geen nieuwe entiteit aangemaakt. De oude kan nog gewoon worden gebruikt.

50005 Entiteit aangemaakt: %1
G → Melding van een succesvol verlopen actie.

50006 Openen logfile: %1

G → Melding dat de logfile is geopend.

50007 Openen uitwisselingsbestand: %1
G → Melding van een succesvol verlopen actie

50008 Openen definitiebestand: %1
G → Melding van een succesvol verlopen actie

50009 Sluiten definitiebestand
G → Melding van een succesvol verlopen actie

50010 Sluiten uitwisselingsbestand
G → Melding van een succesvol verlopen actie

50011 Sluiten logfile
G → Melding van een succesvol verlopen actie

50012 Van entiteit %1 is attribuut: %2 gelezen
G → Melding van een succesvol verlopen actie

50013 Tijdstempel aangepast: %1
G → Melding van een succesvol verlopen actie

50014 Van entiteit %1 is attribuut %2 geschreven
G → Melding van een succesvol verlopen actie

50015 Nieuw record aangemaakt voor entiteit %1
G → Melding van een succesvol verlopen actie

50016 Bestand %1 niet gevonden; Checksum niet uitgevoerd
G → Controleer de bestanden op de schijf.

50017 I/O error bij openen van bestand %1 ; Checksum niet uitgevoerd
G → Controleer de bestanden op de schijf.

8.3 Foutmeldingen Stekkerdoos Water

-50000 Status %1 voor uitwisselingsbestand %2 is niet correct
G → Geef status "NEW", "UPDATE" of "READ"

-49999 vervallen

-49998 Bestand %1 niet gevonden
G → Het bestand moet aanwezig zijn. Controleer dit

-49997 I/O error bij openen/sluiten van bestand %1
G → Controleer het betreffende bestand

-49996 Blanco sleutel is niet toegestaan
G → Vervallen

-49995 Eerst routine RD_KEY gebruiken
G → De entiteit-pointer is nog niet geinitialiseerd.

-49994 Eerst routine WR_KEY gebruiken
G → De entiteit-pointer is nog niet geinitialiseerd.

-49993 Maximum aantal entiteiten bereikt
G → Vervallen

-49992 Attribuutnaam van waardereeks %1 is blanco
G → Een blanco attribuut is hier niet toegestaan.

-49991 Attribuut %1 is een sleutel
G → Sleutelwaarden mogen niet worden gewijzigd. Maak zonodig een nieuw record aan.

-49990 Entiteit %1 is niet aanwezig
G → Vervallen

-49989 Attribuut %1 van entiteit %2 bestaat niet
G → Controleer de stekkerprogrammatuur en vergelijk met de beide stuurbestanden

-49988 Uitwisselingsbestand %1 is al geopend
G → Controleer de stekkerprogrammatuur.

-49987 Datafile is geopend voor uitsluitend lezen
G → Dit is een melding en geen fout. De routines WR* werken niet.

-49986 In entiteit %1 heeft attribuut %2 geen waarde
G → Er is een leesopdracht gedaan van een waarde die op "unused" is geinitialiseerd. Deze waarde moet eerst een echte inhoud krijgen voordat deze kan worden gelezen.

-49985 Error tijdens ophalen systeemtijd
G → Fortran fout in routine WRCRNT

-49984 Opgegeven referentie voor entiteit %1 niet gevonden voor waardereeks %2
G → Controleer of de opgegeven referentieentiteit in het databestand aanwezig is.

-49983 Error bij wegschrijven waardereeks %1 , attribuut %2 ; dimensie is groter dan 1
G → Voor het wegschrijven/lezen van een meerdimensionale waardereeksen moeten de routines WRNDWR en RDNDWR worden gebruikt

-49982 Indices bij wegschrijven waardereeks %1 , attribuut %2 zijn niet correct
G → Controleer of de indices wel liggen binnen het gevulde deel van het bestand.

-49981 Waardereeks %1 is gedefinieerd als 1-dimensionaal
G → Voor het wegschrijven/lezen van een meerdimensionale waardereeks moeten de routines WRNDWR en RDNDWR worden gebruikt

-49980 Aantal dimensies klopt niet met definitie van waardereeks %1
G → Vergelijk de stekkersoftware met de beide stuurbestanden

-49979 Opgegeven attributen voor entiteit %1 zijn niet allemaal sleutels
G → Er zijn sleutels opgegeven die volgens het stuurbestand geen sleutels zijn.

-49978 Meerdere keren dezelfde sleutel opgegeven
G → Een sleutel mag niet vaker dan eenmaal voorkomen. Pas de stekkersofware aan!

-49977 Aantal opgegeven sleutels voor entiteit %1 is niet correct

G → Alle sleutels moeten worden opgegeven.

-49976 Lengte van array om %1 op te slaan is onvoldoende
G → Vergroot de buffer waarin de sleutels moeten worden gelezen

-49975 Attribuut %1 is geen sleutel van entiteit %2
G → vervallen

-49974 Verkeerde sleutel opgegeven voor entiteit %1
G → De opgegeven sleutelnaam is geen sleutelattribuut. Controleer de stekkersoftware en vergelijk met het stuurbestand.

-49973 De opgegeven sleutelnaam %1 is niet van het type CHARACTER
G → Sleutels moeten van het type CHARACTER zijn. Controleer het stuurbestand.

-49972 Groepnaam definitie ongelijk data: %1 , %2
B → Fortran fout in routine INQENT

-49971 Verkeerde status in subroutine UWBOMS
B → Fortran fout in routine UWBOMS

-49970 Verkeerde status in subroutine CHKID
B → Fortran fout in routine CHKID.

-49969 Te veel sleutels opgegeven voor entiteit %1
G → Er zijn meer sleutels opgegeven dan de Stekkerdoos Water aan kan. Zie variabele MXNSLT in ST_Parms.INC

-49968 Opgegeven entiteittype %1 is geen waardereeksdefinitie
G → De opgegeven entiteit staat in het stuurbestand niet als waardereeks gedefinieerd.

-49967 Waardereeks %1 heeft geen attribuut %2
G → Vervallen

-49966 Referentieattribuut %1 is een sleutel. Dit is niet toegestaan
→ Sleutelattributen mogen niet worden aangepast.

-49965 Checksum van definitiebestand %1 klopt niet
G → Het definitiebestand is verschillend aan het definitiebestand bij het aanmaken van het databestand, en daardoor onbetrouwbaar. Vraag een nieuwe set van bestanden van de toeleverancier. (.def en .dat en .chk)

-49964 Checksum van uitwisselingsbestand %1 klopt niet
→ Het databestand is verschillend aan het databestand bij het aanmaken, en daardoor onbetrouwbaar. Vraag een nieuwe set van bestanden van de toeleverancier. (.def en .dat en .chk)

-49963 Wegens checksum fout worden de bestanden read only geopend
G → Bij een checksum fout worden de bestanden "read only" geopend, en kan voor eigen risico nog informatie gelezen worden van het databestand.

8.4 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.

9 Voorbeelden voor stekkers

9.1 Fortran stekker

9.1.1 Voorbeeld programma 1

```
PROGRAM vb01

C   Testprogramma voor de schrijf- en leesfuncties van de entiteiten.

C   Stekkerdoos declaraties en common blocks
    INCLUDE 'ST_COMMN.INC'

C   Stekkerdoos interface definities
    INCLUDE 'ST_INTRF.INC'

C   Declaraties van de GW'96 attributen (gegenereerd met CREDEF)
    INCLUDE 'ATTDEF.INC'

    CHARACTER
C                   Naam uitwisselingsbestand
    *       UWBNAM*13,
C                   Sleutelnamen
    *       SLTNMS(MXNSLT)*(MXSLEN),
C                   Sleutelwaarden
    *       SLTWRD(MXNSLT)*(MXAVLN)

c
    INTEGER
    *       ERROR

C   Bepaal de namen van het uitwisselingsbestand en het definitiebestand
    UWBNAM = 'uwb'
    OPEN (9,FILE='VB01.OUT')

C   Open het uitwisselingsbestand (verwijder een eventueel bestaand uitwisselingsbestand)
    CALL OPNUWB(ERROR, UWBNAM, 'NEW', 'voorbeeld 1')
    IF ( ERROR .LT. 0 ) GOTO 900

C   Maak een entiteit van het entiteittype KST
    CALL MAKENT(ERROR,'KST')
    IF ( ERROR .LT. 0 ) GOTO 900

C   Definieer de sleutelnamen en waarden van het eerste object
    SLTNMS(1) = 'KWKIDENT'
    SLTNMS(2) = 'LBISTATG'
    SLTWRD(1) = 'stuw 1'
    SLTWRD(2) = 'a1'

C   Schrijf het eerste object en vul tevens attribuut KSTSOORT
    CALL WR_KEY (ERROR,'KST',2,SLTNMS,SLTWRD,'kstsoort','s1')
    IF ( ERROR .LT. 0 ) GOTO 900

C   Schrijf attributen KSTAANT en WASWORDT naar het eerste object
    CALL WRCRNT (ERROR,'KST','KSTAANT',4)
```

```
      IF ( ERROR .LT. 0 ) GOTO 900
      CALL WRCRNT (ERROR,'KST','WASWORDT','NEW')
      IF ( ERROR .LT. 0 ) GOTO 900

C     Schrijf het tweede object (de namen van de sleutels blijven ongewijzigd)
      SLTWRD(1) = 'stuw 2'
      SLTWRD(2) = 'a2'
      CALL WR_KEY (ERROR,'KST',2,SLTNMS,SLTWRD,'kstsoort','s2')
      IF ( ERROR .LT. 0 ) GOTO 900

C     Schrijf enkele attributen naar het tweede object
      CALL WRCRNT (ERROR,'KST','KSTAANT',5)
      IF ( ERROR .LT. 0 ) GOTO 900
      CALL WRCRNT (ERROR,'KST','WASWORDT','UPDATE')
      IF ( ERROR .LT. 0 ) GOTO 900

C     Schrijf het derde object en vul attribuut WASWORDT
      SLTWRD(1) = 'stuw 2'
      SLTWRD(2) = 'a1'
      CALL WR_KEY (ERROR,'KST',2,SLTNMS,SLTWRD,'WASWORDT','OLD')
      IF ( ERROR .LT. 0 ) GOTO 900

C     Schrijf enkele attributen naar het derde object
      CALL WRCRNT (ERROR,'KST','KSTAANT',5)
      IF ( ERROR .LT. 0 ) GOTO 900
      CALL WRCRNT (ERROR,'KST','KSTSOORT','s3')
      IF ( ERROR .LT. 0 ) GOTO 900

C     Lees attribuut KSTSOORT van het eerste object (stuw met soort s1)
      CALL RD_KEY (ERROR,'KST',0,SLTNMS,SLTWRD,'KSTSOORT',KSTSOORT)
      IF ( ERROR .LT. 0 ) GOTO 900

C     Lees attribuut LBISTATG (dit is een sleutelattribuut) uit het eerste object
      CALL RDCRNT( ERROR,'KST','LBISTATG',SLTWRD(2))
      IF ( ERROR .LT. 0 ) GOTO 900

C     Zoek het volgende record met dezelfde waarde voor LBISTATG
      CALL NXTOBJ( ERROR,'KST',1,SLTNMS(2),SLTWRD(2))
      IF ( ERROR .LT. 0 ) GOTO 900

C     Lees attribuut KSTSOORT van dit gevonden object (het derde object)
      CALL RDCRNT( ERROR,'KST','KSTSOORT',KSTSOORT)
      IF ( ERROR .LT. 0 ) GOTO 900

C     Schrijf de gevonden waarde van KSTSOORT naar het scherm
      WRITE (9,*) 'KSTSOORT = ', KSTSOORT

C     Lees nu de soort van object van het tweede object
      SLTWRD(1) = 'stuw 2'
      SLTWRD(2) = 'a2'
      CALL RD_KEY( ERROR,'KST',2,SLTNMS,SLTWRD,'KSTSOORT',KSTSOORT)

C     Lees nu attribuut KSTAANT van hetzelfde object
      CALL RDCRNT( ERROR,'KST','KSTAANT',KSTAANT)
      IF ( ERROR .LT. 0 ) GOTO 900

C     Schrijf attribuut KSTSOORT en KSTAANT naar het scherm
      WRITE (9,*) 'KSTSOORT= ',KSTSOORT
      WRITE (9,*) 'KSTAANT = ',KSTAANT


C     Rapporteer een eventuele fout
```

```
900 IF ( ERROR .NE. 0 ) THEN
        WRITE (9,*) 'Programma is fout! ERROR = ', ERROR
    ELSE
        WRITE (9,*) 'Programma is klaar'
        CALL CLSUWB( ERROR )
    ENDIF


    PAUSE
    END
```

9.1.2 Uitvoer programma 1

```
KSTSOORT = s3__
KSTSOORT= s2__
KSTAANT =          5
Programma is klaar
```

NB De '__' tekens worden veroorzaakt doordat slechts 2 characters worden meegegeven aan de functies WRCRNT en WR_KEY. Dit kan worden vermeden door de waarde eerst aan een 'string-variabele' mee te geven en vervolgens deze variabele mee te geven aan de betreffende functie

9.1.3 Voorbeeld programma 2

```
C
C   Voorbeeldprogramma met waardereeks-functies
C
      PROGRAM vb02

C   Stekkerdoos declaraties en common blocks
      INCLUDE 'ST_COMMN.INC'

C   Stekkerdoos interface definities
      INCLUDE 'ST_INTRF.INC'

C   Gebruik geen declaraties van de GW'96 attributen (gegenereerd met CREDEF)
C   INCLUDE 'ATTDEF.INC'

      CHARACTER
C           Naam uitwisselingsbestand
     *   UWBNAM*13,
C           Sleutelnamen
     *   SLTNMS(MXNSLT)*(MXSLEN),
C           Sleutelwaarden
     *   SLTWRD(MXNSLT)*(MXAVLN)

c
      INTEGER
     *        ERROR,
     *        I,Interpolatie(1000)

      Double Precision
     *        x(1000),y(1000)

      Character
     *        NAAM(1000)*12,
     *        KSTSOORT*4,
     *        REF_WRDR*(MXELEN)


C   Bepaal de namen van het uitwisselingsbestand en het definitiebestand
      UWBNAM = 'uwb.dat'
      OPEN (9,FILE='VB02.OUT')

C   Open het uitwisselingsbestand (verwijder een eventueel bestaand uitwisselingsbestand)
      CALL OPNUWB(ERROR, UWBNAM,  'NEW', 'voorbeeld 2')
      IF ( ERROR .LT. 0 ) GOTO 900

C   Maak een entiteit van het entiteittype KST
      CALL MAKENT(ERROR,'KST')
      IF ( ERROR .LT. 0 ) GOTO 900
```

```
C   Definieer de sleutelnamen en waarden van het eerste object
    SLTNMS(1) = 'KWKIDENT'
    SLTNMS(2) = 'LBISTATG'
    SLTWRD(1) = 'stuw 1'
    SLTWRD(2) = 'a1'

C   Schrijf het eerste object en vul tevens attribuut KSTSOORT
    CALL WR_KEY (ERROR,'KST',2,SLTNMS,SLTWRD,'kstsoort','s1')
    IF ( ERROR .LT. 0 ) GOTO 900

C   Schrijf het tweede object (de namen van de sleutels blijven ongewijzigd)
    SLTWRD(1) = 'stuw 2'
    SLTWRD(2) = 'a2'
    KSTSOORT  = 's2'
    CALL WR_KEY (ERROR,'KST',2,SLTNMS,SLTWRD,'kstsoort',KSTSOORT)
    IF ( ERROR .LT. 0 ) GOTO 900

C   Schrijf het derde object en vul attribuut WASWORDT en KSTSOORT
    SLTWRD(1) = 'stuw 2'
    SLTWRD(2) = 'a1'
    CALL WR_KEY (ERROR,'KST',2,SLTNMS,SLTWRD,'WASWORDT','OLD')
    CALL WRCRNT (ERROR,'KST','kstsoort','s3')
    IF ( ERROR .LT. 0 ) GOTO 900

C   Vul een waardereeks met waarden
    DO I = 1,1000
        x(i) = DBLE(i)
        y(i) = DBLE(i/2)
        WRITE (NAAM(i),"('naam-',I4)") I
        Interpolatie(i)=MOD(I,2)
    ENDDO

C   Maak een waardereeks in het uitwisselingsbestand met een referentie in de tweede stuw
    SLTWRD(1) = 'stuw 2'
    SLTWRD(2) = 'a2'
    CALL MAKWRD(ERROR, 'WAAARDEREEKS','WAARDEREEKS_1',
    +           'KST','REF_WRDR',2,SLTNMS,SLTWRD)
    IF ( ERROR .LT. 0 ) GOTO 900

C   Schrijf de waardereeksen naar het uitwisselingsbestand
    CALL WR1DWR(ERROR,'Waardereeks_1','X',1,1000,1,X)
    IF ( ERROR .LT. 0 ) GOTO 900
    CALL WR1DWR(ERROR,'Waardereeks_1','Y',1,1000,1,Y)
    IF ( ERROR .LT. 0 ) GOTO 900
    CALL WR1DWR(ERROR,'Waardereeks_1','NAAM',1,1000,1,NAAM)
    IF ( ERROR .LT. 0 ) GOTO 900
    CALL WR1DWR(ERROR,'Waardereeks_1','Interpol',1,1000,1,
    +                                                  Interpolatie)
    IF ( ERROR .LT. 0 ) GOTO 900


C   Lees attributen KSTSOORT en REF_WRDR van het eerste object (stuw met soort s1)
    KSTSOORT = ''
    CALL RD_KEY (ERROR,'KST',0,SLTNMS,SLTWRD,'KSTSOORT',KSTSOORT)
```

```
      WRITE (9,*) 'ERROR = ', ERROR, ' KSTSOORT = ',KSTSOORT
      IF ( ERROR .LT. 0 ) GOTO 900
      REF_WRDR = ''
      CALL RDCRNT (ERROR,'KST','REF_WRDR',REF_WRDR)
      WRITE (9,*) 'ERROR = ', ERROR, ' REF_WRDR = ',REF_WRDR
      IF (( ERROR .LT. 0 ).AND.(ERROR .NE. -49986)) GOTO 900

C     Lees deze attributen van het eerstvolgende object (stuw met soort s2)
      CALL NXTOBJ(ERROR,'KST',0,SLTNMS,SLTWRD)
      IF ( ERROR .LT. 0 ) GOTO 900
      KSTSOORT = ''
      CALL RDCRNT(ERROR,'KST','KSTSOORT',KSTSOORT)
      WRITE (9,*) 'ERROR = ', ERROR, ' KSTSOORT = ',KSTSOORT
      IF ( ERROR .LT. 0 ) GOTO 900
      REF_WRDR = ''
      CALL RDCRNT(ERROR,'KST','REF_WRDR',REF_WRDR)
      WRITE (9,*) 'ERROR = ', ERROR, '  REF_WRDR = ',REF_WRDR
      IF (( ERROR .LT. 0 ).AND.(ERROR .NE. -49986)) GOTO 900

C     Lees deze attributen van het eerstvolgende object (stuw met soort s3)
      CALL NXTOBJ(ERROR,'KST',0,SLTNMS,SLTWRD)
      IF ( ERROR .LT. 0 ) GOTO 900
      KSTSOORT = ''
      CALL RDCRNT(ERROR,'KST','KSTSOORT',KSTSOORT)
      WRITE (9,*) 'ERROR = ', ERROR, ' KSTSOORT = ',KSTSOORT
      IF ( ERROR .LT. 0 ) GOTO 900
      REF_WRDR = ''
      CALL RDCRNT(ERROR,'KST','REF_WRDR',REF_WRDR)
      WRITE (9,*) 'ERROR = ', ERROR, ' REF_WRDR = ',REF_WRDR
      IF (( ERROR .LT. 0 ).AND.(ERROR .NE. -49986)) GOTO 900

C     Initialiseer de arrays
      DO I=1,1000
         NAAM(I) = ''
         X(i) = 0
         Y(I) = 0
         Interpolatie = -1
      ENDDO

C     Lees de positienamen uit de waardereeks
      CALL RD1DWR (ERROR,'Waardereeks_1','NAAM',1,1000,1,12000,NAAM)
      IF ( ERROR .LT. 0 ) GOTO 900
      CALL RD1DWR (ERROR,'WAARDEREEKS_1','X',1,1000,1,8000,X)
      IF ( ERROR .LT. 0 ) GOTO 900
      CALL RD1DWR (ERROR,'waardereeks_1','Y',1,1000,1,8000,Y)
      IF ( ERROR .LT. 0 ) GOTO 900
      CALL RD1DWR (ERROR,'waardereeks_1','interpol',1,1000,1,8000,
     *                                                  Interpolatie)
      IF ( ERROR .LT. 0 ) GOTO 900

C     Schrijf de resultaten naar het scherm
      DO I = 1,1000
         WRITE (9,*) Naam(I),X(i),Y(i),Interpolatie(i)
      ENDDO
```

```
C   Rapporteer een eventuele fout
C   Het nummer -49986 betekent dat een waarde nog niet is ingevuld.
C   Dit is in deze situatie geen fout!
900 IF (ERROR.EQ.-49986) ERROR = 0

    IF ( ERROR .LT. 0 ) THEN
        WRITE (9,*) 'Programma is fout! ERROR = ', ERROR
    ELSE
        WRITE (9,*) 'Programma is klaar'
        CALL CLSUWB( ERROR )
    ENDIF

    PAUSE
    END
```

9.1.4 Uitvoer programma 2

```
ERROR =           0 KSTSOORT = s1__
ERROR =      -49986 REF_WRDR = _____________
ERROR =           0 KSTSOORT = s2
ERROR =           0  REF_WRDR = WAARDEREEKS_1
ERROR =           0 KSTSOORT = s3__
ERROR =      -49986 REF_WRDR = _____________
naam-   1          1.000000000000000   0.000000000000000E+000          1
naam-   2          2.000000000000000       1.000000000000000          0
naam-   3          3.000000000000000       1.000000000000000          1
naam-   4          4.000000000000000       2.000000000000000          0
naam-   5          5.000000000000000       2.000000000000000          1
... ...
naam- 989        989.000000000000000     494.000000000000000          1
naam- 990        990.000000000000000     495.000000000000000          0
naam- 991        991.000000000000000     495.000000000000000          1
naam- 992        992.000000000000000     496.000000000000000          0
naam- 993        993.000000000000000     496.000000000000000          1
naam- 994        994.000000000000000     497.000000000000000          0
naam- 995        995.000000000000000     497.000000000000000          1
naam- 996        996.000000000000000     498.000000000000000          0
naam- 997        997.000000000000000     498.000000000000000          1
naam- 998        998.000000000000000     499.000000000000000          0
naam- 999        999.000000000000000     499.000000000000000          1
naam-1000       1000.000000000000000     500.000000000000000          0
Programma is klaar
```

9.1.5 Voorbeeld programma 3

```
C
C  Voorbeeldprogramma met opvraagfuncties

      PROGRAM vb03


C  Stekkerdoos declaraties en common blocks
      INCLUDE 'ST_COMMN.INC'

C  Stekkerdoos interface definities
      INCLUDE 'ST_INTRF.INC'

C  Gebruik geen declaraties van de GW'96 attributen (gegenereerd met CREDEF)
C  INCLUDE 'ATTDEF.INC'

      CHARACTER
C                 Naam uitwisselingsbestand
     *        UWBNAM*13
c
      INTEGER
     *            I,
     *            ERROR,
     *            AANTEN,
     *            AANTWR,
     *            NREC,
     *            NATT,
     *            NSLT,
     *            NDIM,
     *            DIMS(5),
     *            NEFLEN

      CHARACTER
     *            OMSUWB*(MXLEN1),
     *            LDATEUWB*(LENDAT),
     *            VERSIEGW*64,
     *            NAAMSDW*64,
     *            STBSTDATE*64,
     *            VERSIEBIL*64,
     *            TOEP*64,
     *            BILBSTDATE*64,
     *            ENTNMS(MXNENT)*(MXELEN),
     *            WRDNMS(MXNRKS)*(MXALEN),
     *            ATTNMS(MXNATT)*(MXALEN),
     *            SLTNMS(MXNSLT)*(MXSLEN),
     *            BILENT*1,
     *            ENTITY*(MXELEN),
     *            BILWRD*1,
     *            REFENT*(MXELEN),
     *            REFATT*(MXALEN),
     *            SLTWRD(MXNSLT)*(MXAVLN),
```

```
*           ATTTYP*16,
*           ATTLEN*16,
*           BILATT*1,
*           NEFTYP*8


C   Bepaal de namen van het uitwisselingsbestand en het definitiebestand
    UWBNAM = 'uwb.dat'
    OPEN (9,File='VB03.OUT')

C   Open het uitwisselingsbestand (verwijder een eventueel bestaand uitwisselingsbestand)
    CALL OPNUWB(ERROR, UWBNAM,  'READ', OMSUWB)
    WRITE (9,*) 'omschrijving van het uitwisselingsbestand = ',OMSUWB
    IF ( ERROR .LT. 0 ) GOTO 900

C   Vraag alle administratieve gegevens van het uitwiosselingsbestand
    CALL INQUWB (ERROR, OMSUWB, LDATEUWB, VERSIEGW, NAAMSDW,
    *                                          STBSTDATE, VERSIEBIL, TOEP,
BILBSTDATE)
    WRITE (9,*) 'ERROR =                ',ERROR
    WRITE (9,*) 'Omschrijving UWB =         ',OMSUWB
    WRITE (9,*) 'datum laatste wijziging uwb = ',LDATEUWB
    WRITE (9,*) 'versie GW-classificatie =    ',VERSIEGW
    WRITE (9,*) 'Naam SDW model =           ',NAAMSDW
    WRITE (9,*) 'Generatiedatum stuurbestand = ',STBSTDATE
    WRITE (9,*) 'Versie bilaterale afspraken = ',VERSIEBIL
    WRITE (9,*) 'Toepassing =               ',TOEP
    WRITE (9,*) 'Datum aanmaak bilateraalbest =',BILBSTDATE
    PAUSE

C   Vraag de namen van de opgeslagen entiteiten en waardereeksen
    AANTEN = MXNENT
    AANTWR = MXNRKS
    CALL INQCNT(ERROR,AANTEN,ENTNMS,AANTWR,WRDNMS)
    WRITE (9,*) 'ERROR =                ',ERROR
    WRITE (9,*) 'aantal entiteiten          ',AANTEN
    WRITE (9,*) 'aantal waardereeksen       ',AANTWR
    WRITE (9,*) '   entiteiten en waardereeksen'
    DO I = 1,MAX(AANTEN,AANTWR)
        WRITE (9,*) I, ' ',ENTNMS(I),WRDNMS(I)
    ENDDO
    PAUSE

C   Vraag de informatie van een entiteit (de eerste)
    NATT = MXNATT
    NSLT = MXNSLT
    CALL INQENT(ERROR, ENTNMS(1), NREC, NATT, ATTNMS, NSLT, SLTNMS,
   *
BILENT)
    WRITE (9,*) 'ERROR =                ',ERROR
    WRITE (9,*) 'aantal objecten =          ',NREC
    WRITE (9,*) 'aantal attributen =        ',NATT
    WRITE (9,*) 'aantal sleutels =          ',NSLT
    WRITE (9,*) 'bilateraal                 ',BILENT
```

```
      DO I=1,MIN(NATT,NSLT)
         WRITE (9,*) I, ' ',ATTNMS(i),SLTNMS(I)
      ENDDO
      DO I = MIN(NATT,NSLT)+1,MAX(NATT,NSLT)
         WRITE (9,*) I, ' ', ATTNMS(I)
      ENDDO
      PAUSE

C     Vraag de informatie van een entiteit (de eerste)
      IF (AANTWR.GT.0) THEN
         NATT =MXNATT
         NSLT = MXNSLT
         CALL INQWRD(ERROR,WRDNMS(1),ENTITY,NDIM,DIMS,NREC,NATT,
     *              ATTNMS,BILWRD,REFENT,REFATT,NSLT,SLTNMS,SLTWRD)
         WRITE (9,*) 'ERROR =                    ',ERROR
         WRITE (9,*) 'naam waardereeks =           ',WRDNMS(1)
         WRITE (9,*) 'naam entiteittype =          ',ENTITY
         WRITE (9,*) 'aantal dimensies =           ',NDIM
         WRITE (9,*) 'in stuurbestand opgegeven dims = ',
     *                                          (DIMS(I),I=1,NDIM-1)
         WRITE (9,*) 'vrije dimensie =           ',DIMS(NDIM)
         WRITE (9,*) 'aantal attributen =        ',NATT
         WRITE (9,*) 'aantal attributen =        ',NATT
         WRITE (9,*) 'bilateraal =              ',BILWRD
         WRITE (9,*) 'naam van de ref. eniteit =    ',REFENT
         WRITE (9,*) 'naam ref. attribuut =       ',REFATT
         WRITE (9,*) 'aantal sleutels ref. ent.=    ',NSLT
         WRITE (9,*) 'attribuutnamen:'
         DO I=1,NATT
            WRITE (9,*) I, ' ',ATTNMS(i)
         ENDDO
         WRITE (9,*) 'sleutelnamen en sleutelwaarden referentie entiteit'
            DO I = 1,NSLT
            WRITE (9,*) I, ' ', SLTNMS(I), TRIM(SLTWRD(I))
         ENDDO
         PAUSE
      ENDIF

C     Vraag de informatie van een attribuut (de tweede van de eerste entiteit)
      CALL INQATT(ERROR, ATTNMS(1), ATTTYP, ATTLEN, BILATT, NEFTYP,
     *                                                      NEFLEN)
      WRITE (9,*) 'ERROR =                  ',ERROR
      WRITE (9,*) 'attribuut =              ',ATTNMS(1)
      WRITE (9,*) 'attribuuttype =           ',ATTTYP
      WRITE (9,*) 'attribuut lengte =        ',ATTLEN
      WRITE (9,*) 'bilateraal =             ',BILATT
      WRITE (9,*) 'Nefis declaratie =         ',NEFTYP
      WRITE (9,*) 'Nefis lengte (in bytes) =    ',NEFLEN
      PAUSE

C     Rapporteer een eventuele fout
C     Het nummer -49986 betekent dat een waarde nog niet is ingevuld.
C     Dit is in deze situatie geen fout!
900   IF (ERROR.EQ.-49986) ERROR = 0
```

```
IF ( ERROR .LT. 0 ) THEN
    WRITE (9,*) 'Programma is fout! ERROR = ', ERROR
    CALL CLSUWB( ERROR)
ELSE
    WRITE (9,*) 'Programma is klaar'
    CALL CLSUWB( ERROR )
ENDIF

PAUSE
END
```

9.1.6 Uitvoer programma 3

Deze uitvoer is gegenereerd met het databestand gegenereerd door programma VB02.

```
omschrijving van het uitwisselingsbestand = voorbeeld 2

ERROR =                                  0
Omschrijving UWB =            voorbeeld 2

datum laatste wijziging uwb = Fri Aug 15 17:37:37 1997
versie GW-classificatie =     1.0

Naam SDW model =              GW96

Generatiedatum stuurbestand = 31/07/1997

Versie bilaterale afspraken = Voorbeeld van een bilateraal stuurbestand met een
 1 dimensionale
Toepassing =                  Voorbeeld 2

Datum aanmaak bilateraalbest =14/08/1997

ERROR =                                  0
aantal entiteiten                        1
aantal waardereeksen                     1
   entiteiten en waardereeksen
         1 KST           WAARDEREEKS_1
ERROR =                                  0
aantal objecten =                        3
aantal attributen =                      3
aantal sleutels =                        2
bilateraal                    J
         1 WASWORDT      KWKIDENT
         2 KSTSOORT      LBISTATG
         3 REF_WRDR
ERROR =                                      0
naam waardereeks =                WAARDEREEKS_1
naam entiteittype =               WAAARDEREEKS
aantal dimensies =                           1
in stuurbestand opgegeven dims =
vrije dimensie =                          1000
aantal attributen =                          4
aantal attributen =                          4
bilateraal =                      J
naam van de ref. eniteit =        KST
naam ref. attribuut =             REF_WRDR
aantal sleutels ref. ent.=                   2
attribuutnamen:
         1 X
         2 Y
         3 NAAM
         4 INTERPOL
sleutelnamen en sleutelwaarden referentie entiteit
         1 KWKIDENT      stuw 2
         2 LBISTATG      a2
ERROR =                                  0
attribuut =                       X
attribuuttype =                   NUMERIEK
attribuut lengte =                12.1
bilateraal =                      J
Nefis declaratie =                REAL
Nefis lengte (in bytes) =                8
```

```
Programma is klaar
```

9.2 C-Stekker

9.2.1 Sources

```
/*          Copyright (C) 1993 Delft Hydraulics

 System     : Stekkerdoos water


 Programmer : Peter van den Bosch
 Part       : Test programma C-interface Stekkerdoos
 */

#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include "st_parms.h"
#include "st_cintf.h"

void UwbGeg( int * error, FILE * fprn );
void EntGeg( int * error, FILE * fprn, int * aanent, char entisys[][MXELEN] );
void EntRec( int * error, FILE * fchk, char * uwbnam,
        int * aanent, char entisys[][MXELEN] );


void main( int argc, char ** argv )
{
 int error;
 int aanent;
 char uwbnam[50];
 char omschr[MXLEN1];
 char entisys[MXNENT][MXELEN];
 char bil[2];
 char attype[17];
 char attlen[17];
 char neftype[17];
 char sltnms[MXNSLT][MXSLEN];
 char sltwrd[MXNSLT][MXAVLN];
 char awaarde[MXAVLN];
 char creeks[50][40][12];

 int i, j, ndim, d_buflen, c_buflen, neflen;

 int start[2], n_stap[2], delta[2];

 double ireeksx[50][40], ireeksy[50][40];
 double oreeksx[50][40], oreeksy[50][40];

 FILE * fchk;
```

```
FILE * fprn;


strcpy( uwbnam, "uwb.dat" );
strcpy( omschr, "" );

fprn = fopen( "vb04.out", "w" );
fchk = fopen( "vb04.dat", "w" );
fprintf( fprn, "Data bestand " );

/* open uitwisselingsbestand */
C_OPNUWB( &error, uwbnam, "NEW", omschr );
if ( error < 0 )
{
   exit(1);
}

/* Maak de entiteit kst */
C_MAKENT( &error, "kst" );
if ( error < 0 )
{
   exit(1);
}
strcpy( sltnms[0], "LBISTATG" );
strcpy( sltnms[1], "KWKIDENT" );
strcpy( sltwrd[0], "a" );
strcpy( sltwrd[1], "a" );

/* Schrijf KSTSOORT naar het uitwisselingsbestand */
C_WR_KEY( &error, "kst", 2, sltnms, sltwrd, "kstsoort", "aa" );
if ( error < 0 )
{
   exit(1);
}

strcpy( sltwrd[1], "b" );
C_WR_KEY( &error, "kst", 2, sltnms, sltwrd, "kstsoort", "bb" );
if ( error < 0 )
{
   exit(1);
}

strcpy( sltwrd[1], "b" );

/* creëer 1-d waardenreeks */
C_MAKWRD( &error, "kstwrd01", "WRD1d", "kst", "kstwrd01", 2, sltnms, sltwrd );
if ( error < 0 )
{
   exit(1);
}

/* creëer 2-d waardenreeks */
C_MAKWRD( &error, "kstwrd02", "WRD0x", "kst", "kstwrd02", 2, sltnms, sltwrd );
if ( error < 0 )
```

```
{
   exit(1);
}
strcpy( awaarde, "" );

/* Lees het referentie attribuut uit KST */
C_RD_KEY( &error, "kst", 2, sltnms, sltwrd, "kstwrd02", awaarde );
if ( error < 0 )
{
   exit(1);
}

for ( i = 0; i < 50; i++ )
{
  for ( j = 0; j < 40; j++ )
     {
        ireeksx[i][j] = i;
              ireeksy[i][j] = j;
              sprintf( creeks[i][j], "pnt %d %d\n", i, j );
     }
}

ndim = 2;
start[0] = 1;
start[1] = 1;
n_stap[0] = 50;
n_stap[1] = 40;
delta[0] = 1;
delta[1] = 1;

/* schrijf 1-d waardenreeks */
C_WR1DWR( &error, "WRD1d", "x", start[0], n_stap[0], delta[0], ireeksy );

/* schrijf 2-d waardenreeks */
C_WRNDWR( &error, "WRD0x", "x", 2, start, n_stap, delta, ireeksx );
C_WRNDWR( &error, "WRD0x", "y", 2, start, n_stap, delta, ireeksy );
C_WRNDWR( &error, "WRD0x", "posbesch", 2, start, n_stap, delta, creeks );
if ( error < 0 )
{
   exit(1);
}

for ( i = 0; i < 50; i++ )
{
  for ( j = 0; j < 40; j++ )
     {
        oreeksx[i][j] = 0;
              oreeksy[i][j] = 0;
              sprintf( creeks[i][j], "  \n" );
     }
}

d_buflen = 2000*8;
c_buflen = 2000*12;
```

```
/* lees 1-d waardenreeks */
C_RD1DWR( &error, "WRD1d", "x", start[0], n_stap[0], delta[0], d_buflen, ireeksx );

/* lees 2-d waardenreeks */
C_RDNDWR( &error, "WRD0x", "x", 2, start, n_stap, delta, d_buflen, oreeksx );
C_RDNDWR( &error, "WRD0x", "y", 2, start, n_stap, delta, d_buflen, oreeksy );
C_RDNDWR( &error, "WRD0x", "posbesch", 2, start, n_stap, delta, c_buflen, creeks );
if ( error < 0 )
{
   exit(1);
}

fprintf( fprn, " 1-d reeks \n" );
for ( i = 0; i < 50; i++ )
{
 fprintf( fprn, " %f \n", ireeksx[0][i] );
}
fprintf( fprn, " 2-d reeks \n" );
for ( i = 0; i < 50; i++ )
{
  for ( j = 0; j < 40; j++ )
     {
        fprintf( fprn, " %f %f %s \n", oreeksx[i][j], oreeksy[i][j], creeks[i][j] );
     }
}

UwbGeg( &error, fprn );
if ( error < 0 )
{
   exit(1);
}


/* print gegevens aanwezige entiteiten */
EntGeg( &error, fprn, &aanent, entisys );
if ( error < 0 )
{
   exit(1);
}

/* attribuutgegevens */
C_INQATT( &error, "kstwrd01", &attype, &attlen, &bil, &neftype,
                                        &neflen);
fprintf( fprn, "Attribuutgegevens          : %s \n", "kstwrd01" );
fprintf( fprn, "type                       : %s \n", attype );
fprintf( fprn, "lengte                     : %s \n", attlen );
fprintf( fprn, "bil                        : %s \n", bil    );
fprintf( fprn, "nefis type                 : %s \n", neftype   );
fprintf( fprn, "nefis lengte               : %d \n", neflen    );


C_CLSUWB( &error );
if ( error < 0 )
```

```
    {
      exit(1);
    }

}

void UwbGeg(
 int  * error,
 FILE * fprn )
{
 char descr[200];
 char lst_date[LENDAT];
 char gw_cls[65];
 char sdw_nam[65];
 char gn_date[65];
 char bil_vs[65];
 char appl[65];
 char bil_date[65];
 char wrd[17];
 char entity[17];
 char bil[2];
 char refent[17];
 char refatt[17];
 char sltnms[MXNSLT][MXSLEN];
 char sltwrd[MXNSLT][MXAVLN];
 char attnms[MXNATT][MXALEN];

 int ndim, nrec, natt, nslt;

 int dims[5];

 /* opvragen en printen algemene gegevens uitwisselingsbestand */

 C_INQUWB( error, descr, lst_date, gw_cls, sdw_nam, gn_date, bil_vs,
        appl, bil_date );
 if ( error < 0 ) return;

 fprintf( fprn, "Data bestand " );
 fprintf( fprn, "Omschrijving : %s \n", descr );
 fprintf( fprn, "Datum laatste wijziging          : %s \n", lst_date );
 fprintf( fprn, "Versie GW_classificatie          : %s \n", gw_cls );
 fprintf( fprn, "Naam SDW-model                   : %s \n", sdw_nam );
 fprintf( fprn, "Datum generatie stuurbestand : %s \n", gn_date );
 fprintf( fprn, "Versie bilaterale afspraken      : %s \n", bil_vs );
 fprintf( fprn, "Toepassing                       : %s \n", appl );
 fprintf( fprn, "Datum bilateraal bestand         : %s \n", bil_date );

 /* opvragen gegevens waardereeks */
 strcpy( wrd , "WRD0x" );
 strcpy( sltnms[0], "" );
 strcpy( sltnms[1], "" );
 strcpy( sltwrd[0], "" );
 strcpy( sltwrd[1], "" );
 natt = MXNATT;
```

```
  nslt = MXNSLT;

  C_INQWRD( &error, wrd, entity, &ndim, dims, &nrec, &natt, attnms, bil,
        refent, refatt, &nslt, sltnms, sltwrd );
  if ( error < 0 ) return;

  fprintf( fprn, "Waardenreeks                    : %s \n", wrd );
  fprintf( fprn, "entity          : %s \n", entity );
  fprintf( fprn, "ndim                            : %d \n", ndim   );
  fprintf( fprn, "dimensions                      : %d %d \n", dims[0], dims[1] );
  fprintf( fprn, "nrec                            : %d \n", nrec   );
  fprintf( fprn, "natt                            : %d \n", natt   );
  fprintf( fprn, "attnms                          : %s \n", attnms[0] );
  fprintf( fprn, "bil                             : %s \n", bil    );
  fprintf( fprn, "refent          : %s \n", refent   );
  fprintf( fprn, "refatt                          : %s \n", refatt    );
  fprintf( fprn, "nslt                            : %d \n", nslt   );
  fprintf( fprn, "sltnms                          : %s \n", sltnms[0] );
  fprintf( fprn, "sltwrd                          : %s \n", sltwrd[0] );

  return;
}

void EntGeg(
  int * error,
  FILE * fprn,
  int * aanent,
  char  entisys[][MXELEN] )
{

  char wrdrks[MXNENT][MXELEN];
  int  aanwrd, i;

  *aanent = MXNENT;
  aanwrd = MXNENT;

  C_INQCNT( error, aanent, entisys, &aanwrd, wrdrks );
  if ( error < 0 )
  {
     exit(1);
  }

  /* schrijf de entiteitsnamen naar de printfile */
  fprintf( fprn, " \n \n " );
  fprintf( fprn, "Namen van de aanwezige entiteiten \n" );
  for ( i=0; i<*aanent; i++ )
  {
     fprintf( fprn, " %d  %s \n", i, entisys[i] );
  }
  /* schrijf de waardenreeksnamen naar de printfile */
  fprintf( fprn, " \n \n " );
  fprintf( fprn, "Namen van de aanwezige waardenreeksen \n" );
  for ( i=0; i<aanwrd; i++ )
  {
```

```
    fprintf( fprn, " %d  %s \n", i, wrdrks[i] );
  }
  return;
}
```

9.2.2 Programma uitvoer

```
Data bestand  1-d reeks
 0.000000
 1.000000
 2.000000
 3.000000
 4.000000
 … …
 34.000000
 35.000000
 36.000000
 37.000000
 38.000000
 39.000000
 0.000000
 1.000000
 2.000000
 3.000000
 4.000000
 5.000000
 6.000000
 7.000000
 8.000000
 9.000000
 2-d reeks
 0.000000 0.000000 pnt 0 0

 0.000000 1.000000 pnt 0 1

 0.000000 2.000000 pnt 0 2

 0.000000 3.000000 pnt 0 3

 0.000000 4.000000 pnt 0 4
 … …

 0.000000 38.000000 pnt 0 38

 0.000000 39.000000 pnt 0 39

 1.000000 0.000000 pnt 1 0

 1.000000 1.000000 pnt 1 1

 1.000000 2.000000 pnt 1 2
 … …
 49.000000 36.000000 pnt 49 36

 49.000000 37.000000 pnt 49 37

 49.000000 38.000000 pnt 49 38

 49.000000 39.000000 pnt 49 39

Data bestand Omschrijving                :
Datum laatste wijziging      : Mon Aug 18 18:14:33 1997
Versie GW_classificatie      : 1.0
Naam SDW-model               : GW96
Datum generatie stuurbestand : 01/05/1997
Versie bilaterale afspraken  : Dit bestand is voor test 3.
Toepassing                   : De toepassing is het testen van de Stekkerdoos
Datum bilateraal bestand     : 24/04/1997
Waardenreeks                 : WRD0x
entity                       : KSTWRD02
ndim                         : 2
dimensions                   : 50 0
nrec                         : 40
natt                         : 4
attnms                       : X
bil                          : J
```

```
refent                          : KST
refatt                          : KSTWRD02
nslt                            : 2
sltnms                          : KWKIDENT
sltwrd                          : b

 Namen van de aanwezige entiteiten
 0  KST

 Namen van de aanwezige waardenreeksen
 0  WRD1D
 1  WRD0X
Attribuutgegevens               : kstwrd01
type                            : Alfanumeriek
lengte                          : 12
bil                             : J
nefis type                      : CHARACT
nefis lengte                    : 12
```

10 Referenties

Ref 1: GW'96	Gegevensmodel Water
Ref 2: SDWrite	SDWrite van SDW® geleverd door Cap Gemini.
Ref 3: Func.Ontw.	Functioneel Ontwerp Stekkerdoos Water; Waterloopkundig Laboratorium; F.A. Douma; augustus 1997
Ref 4: Func.Ontw.	Technische documentatie Stekkerdoos Water; Waterloopkundig Laboratorium; F.A. Douma; augustus 1997

Stekkerdoos
Water

Functioneel Ontwerp

Inhoud

1 Samenvatting **1-1**

2 Inleiding **2-2**

2.1 Opdrachtbeschrijving 2-2

2.2 Probleemstelling 2-2

2.3 Soorten gegevensuitwisseling 2-3

3 Uitgangspunten en randvoorwaarden **3-1**

3.1 Randvoorwaarden uit de offerte aanvraag 3-1

3.2 Additionele uitgangspunten/randvoorwaarden 3-6

4 Gewenst systeem **4-1**

4.1 Globale systeembeschrijving 4-1

4.2 Systeem context 4-1

5 Functioneel ontwerp **5-1**

5.1 Functioneel model 5-1

5.2 Gegevensmodel 5-3

5.3 Waardereeksen 5-4

5.3.1 Algemeen 5-4
5.3.2 Waardereeksen in de stekkerdoos 5-4

5.4 Externe interfaces 5-5

5.5 Gebruikersinteractie 5-5

5.6 Technische eisen/ontwerp-beslissingen 5-5

6 Organisatorische aspecten **6-1**

6.1 Betrokkenheid van de gebruikers bij projectontwikkeling: 6-1

6.2 Afstemming met stekkerbouwers 6-1

6.3 Testen 6-1

6.4 Beheer en onderhoud 6-1

7 Beschrijving stekkerdoos bibliotheek **7-1**

8 Beschrijving van de stuurtabellen **8-1**

9 Beschrijving van Nefis 9-1

1 Samenvatting

Het doel van dit project is om een faciliteit te bieden welke organisaties in staat stelt om op basis van de Gegevensstandaard WATER 1996 (GW'96) op eenvoudige en gestandaardiseerde manier gegevens uit te wisselen. Deze faciliteit (Stekkerdoos genoemd) moet aangesloten kunnen worden bij de gangbare en relevante toepassingen van de NEN-1878 en de NEN-3610 (Vastgoed) en geen verhindering vormen voor de overgang naar de in ontwikkeling zijnde Europese normen voor vastgoed beheer.

Het voor u liggende document beschrijft de functionaliteit en de technische opzet van de Stekkerdoos. In deze stekkerdoos zitten functies voor het uitwisselen van GW'96 geclassificeerde gegevens, maar ook voor het uitwisselen van meetreeksen. Bovendien kunnen aanvullende gegevens-classificatie-afspraken worden gemaakt en gegevens volgens deze classificatie worden geschreven en gelezen.

Voordat de gegevens kunnen worden geschreven moeten deze wel eerst worden geconverteerd naar de GW-96 classificatie of de bilaterale afspraken. Deze conversie vindt plaats in de te bouwen stekkers. Dit project levert ook faciliteiten die de stekkerbouwer ondersteunen voor het bouwen van deze conversies.

De Stekkerdoos wordt gerealiseerd door een "high-level-schil" te bouwen om het WL-produkt NEFIS. De uitwisseling vindt plaats door middel van platformonafhankelijke NEFIS-bestanden. Voor de NEN-1878 toepassingen wordt parallel een onderzoek uitgevoerd door HKV Lijn in Water.

2 Inleiding

2.1 Opdrachtbeschrijving

Het doel van dit project is om een faciliteit te bieden welke organisaties in staat stelt om op basis van de Gegevensstandaard WATER 1996 (GW'96) op eenvoudige en gestandaardiseerde manier gegevens uit te wisselen. Deze faciliteit moet aangesloten kunnen worden bij de gangbare en relevante toepassing van de NEN-1878 en de NEN-3610 (Vastgoed) en geen verhindering vormen voor de overgang naar de in ontwikkeling zijnde Europese normen voor vastgoed beheer.

Er is reeds een functioneel model opgesteld voor een stekkerdoos gebaseerd op NEN-1878 bestanden. Uit dit ontwerp blijkt dat een generiek systeem voor het lezen en schrijven van willekeurige NEN-1878 bestanden dermate complex wordt dat onzeker is of dit systeem in de praktijk nog wel zal worden gebruikt. Bovendien zijn de realisatiekosten veel hoger dan de aanvankelijke begroting.
Dit alles was voor de opdrachtgever de aanleiding om in de offerteaanvraag te vragen om een breder onderzoek met betrekking tot alternatieven. Dit onderzoek heeft er uiteindelijk toe geleid dat opdracht is verleend voor de realisatie van een Stekkerdoos Water op basis van NEFIS voor de uitwisseling van GW'96 gegevens en waardereeksen. Daarnaast is door HKV Lijn in Water een inventarisatie uitgevoerd van wenselijkheid en de mogelijkheden voor een op NEN-1878 gebaseerde gegevensuitwisseling voor met name geometrie en cartografie. De resultaten van dit onderzoek zullen buiten bezwaar van dit project worden verwerkt.

Dit functioneel ontwerp is opgesteld door Ing. F.A. Douma, inhoudelijk gereviewed door Ing. M. de Rover en vrijgegeven door Dr. G.K. Verboom.

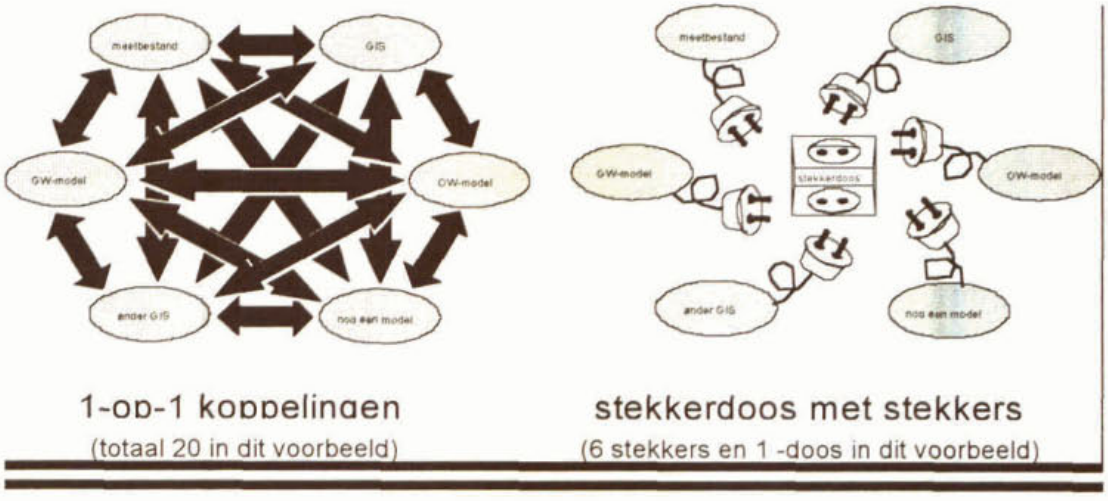
Belangrijkste wijzigingen t.o.v. de vorige versie

- Een aantal kleine redactionele wijzigingen.
- Verplaatsing van de detailbeschrijving van de Stekkerdoos-bibliotheek en de stuurbestanden naar de Gebruikershandleiding.

2.2 Probleemstelling

Gegevensuitwisseling tussen organisaties en informatiesystemen verloopt veel efficiënter wanneer betrokken partijen hun informatievoorziening geconformeerd hebben aan een gemeenschappelijk uitgangspunt. Dit gemeenschappelijke uitgangspunt is het classificatiestelsel GW'96. Door dit gemeenschappelijk classificaties stelsel wordt gerealiseerd dat grootheden en termen op een eenduidig beschreven manier worden opgeslagen in uitwisselingsbestanden. Hierdoor is het mogelijk om informatiesystemen te realiseren die op elkaar aansluitbaar en uitwisselbaar zijn. Dit werkt kostenbesparend. Bovendien wordt het voor leveranciers aantrekkelijk om systemen te ontwikkelen die op de stekkerdoos gebaseerd zijn, omdat deze breder inzetbaar zijn.

Voor de uitwisseling van gegevens tussen bestaande applicaties is steeds 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 applicatie. Dit wordt grafisch in beeld gebracht in figuur 1.



Figuur 1, functie van de stekkerdoos

2.3 Soorten gegevensuitwisseling

We onderscheiden vier verschillende soorten gegevensuitwisseling:

1. Off-line gegevensuitwisseling tussen organisaties.
   Hierbij wisselen organisaties gegevensbestanden uit. Dit gebeurt in de huidige situatie zeer frequent.
2. On-line gegevensuitwisseling tussen organisaties.
   Verschillende organisaties werken op dezelfde database of hebben direct toegang tot elkaars database. Deze vorm van gegevensuitwisseling wordt in dit project niet beschouwd.
3. Off-line gegevensuitwisseling tussen informatiesystemen.
   Hierbij levert het ene systeem een gegevensbestand aan dat ingelezen wordt voor verwerking door het volgende programma. Een voorbeeld hiervan is het ophalen van tijdreeksen uit de database (basisregistratie) die als randvoorwaarden worden toegepast binnen een rekenmodel. De uitkomsten van het rekenmodel worden vervolgens overgebracht naar een GIS-applicatie voor presentatie en analyse. Dit soort verwerkingsslagen en bijkomende gegevensuitwisseling zullen vaak door één persoon worden uitgevoerd. De off-line gegevensuitwisseling tussen informatiesystemen is het onderwerp geweest van het SUF-OW-project [2].
4. On-line gegevensuitwisseling tussen informatiesystemen.
   Bij deze vorm van gegevensuitwisseling versmelten de verschillende informatiesystemen als het ware. Een voorbeeld is de uitwisseling tussen een oppervlaktewater-rekenmodel en een afvoer/regenmodel waarbij na iedere tijdstap (of een aantal hiervan) rekenresultaten worden doorgegeven van het ene model naar het andere en omgekeerd. Deze soort van gegevensuitwisseling neemt in belang toe en wordt steeds vaker toegepast.

In dit project richten we ons op de punten 1, 3 en 4: off-line gegevensuitwisseling en on-line uitwisseling tussen informatiesystemen, waarbij voor punt 4 geldt dat dit aspect wordt onderzocht en vooralsnog buiten dit functioneel ontwerp valt).

3 Uitgangspunten en randvoorwaarden

3.1 Randvoorwaarden uit de offerte aanvraag

In deze paragraaf zijn de functionele en technische randvoorwaarden van de offerteaanvraag overgenomen. Bij elk punt is kort aangegeven hoe dit punt in de stekkerdoos is opgenomen.

2 Classificatiestelsels
Voor de uitwisseling zal de GW'96 het standaard te gebruiken classificatiestelsel zijn voor administratieve gegevens. Daarnaast moeten de routines in staat zijn om informatie die daarin niet blijkt te zijn opgenomen, te putten uit een 2e stelsel. Zo worden ook bilaterale afspraken tussen twee uitwisselende organisaties ondersteund.

Stekkerdoos:

1. Vanuit GW'96 wordt een stuurtabel gegenereerd. Deze stuurtabel is de schakel tussen GW'96 en de Stekkerdoos-bibliotheek. Naast deze GW'96 stuurtabel kan tevens een eigen stuurtabel worden aangemaakt, waarin bilaterale afspraken en afwijkingen van GW'96 kunnen worden vastgelegd. Deze tweede stuurtabel heeft dezelfde structuur en formaat als de GW'96-stuurtabel. Bij tegenstrijdigheden tussen de beide stuurtabellen krijgen de bilaterale afspraken de prioriteit boven de definities van GW'96.

Gebruik
De routines worden opgeleverd als routine bibliotheek en kunnen worden aangeroepen vanuit een C., C++ en Fortran ontwikkelomgeving.

Stekkerdoos:

1. De stekkerbouwer kan vanuit Fortran, C of C++ programma's de stekkerdoos bibliotheek aanroepen.

Verwerking
Veelal zal een batch-gewijze verwerking van het in- en uitpakproces gangbaar zijn. Het moet echter mogelijk zijn de routines in te zetten wanneer vanuit een applicatie direct een (deel van een) uitwisselingsbestand benaderd moet worden voor lezen of schrijven.

Stekkerdoos:

1. De Stekkerdoos-bibliotheek is geschikt voor batchprogramma's (off-line). De bibliotheek schrijft de opgetreden fouten in een "logfile" en geeft een foutcode aan de stekker door aan de hand waarvan de stekkerbouwer zelf kan beslissen hoe een opgetreden fout wordt afgehandeld.
2. Door het gebruik van NEFIS is het mogelijk om gegevens in een willekeurige volgorde naar het uitwisselingsbestand te schrijven en te lezen. Hierdoor is de stekkerdoos bij uitstek geschikt om te worden ingezet in applicaties die het uitwisselingsbestand direct benaderen (on-line).

Waardereeksen
Het uitwisselen van waardereeksen moet worden ondersteund, ongeacht of deze materie onderdeel vormt van de GW'96.

Stekkerdoos:

1. In GW'96 is een definitie voor waardereeksen opgenomen, die wordt meegenomen in de stekkerdoos. (Deze definitie is voor GW'96 overgenomen van SUF-OW.) Afwijkende structuren van waardereeksen kunnen in het stuurbestand met bilaterale afspraken worden opgenomen. In dit ontwerp zijn speciale functies voor het lezen en schrijven van meerdimensionale waardereeksen.

Uitwisselingsnorm

Het te gebruiken uitwisselingsformaat is vrij en mag afhankelijk zijn van de uit te wisselen gegevens. Maar de standaard die gehanteerd wordt, moet wel gebruikt worden zonder afwijkende/aanvullende afspraken. Voor de stekkerbouwer moet het gebruikte uitwisselingsformaat transparant zijn.

Stekkerdoos:

1. Er is gekozen voor een NEFIS formaat als uitwisselingsformaat. Momenteel loopt een onderzoek naar het gebruik van een NEN-1878 formaat en de noodzaak en mogelijke inpassing daarvan in de stekkerdoos. De consequenties daarvan zullen in een aanvullende opdracht moeten worden verwerkt.
2. Er wordt om NEFIS een "high-level-schil" gerealiseerd. De stekkerbouwer maakt alleen gebruik van deze high-level-schil. Deze high-level-schil biedt slechts een subset van de standaard NEFIS functionaliteit. Voor deze selectie is gekozen omdat het aanbieden van de gehele NEFIS-functionaliteit de stekkerdoos-interface onnodig complex maakt. (Elk uitwisselingsbestand dat door de high-level-schil is aangemaakt kan met standaard NEFIS functies kan worden gelezen en geschreven, maar omgekeerd kan niet elk willekeurig aangemaakt NEFIS bestand door de high-level-schil worden gelezen en geschreven, omdat de stekkerdoos extra (GW'96 gerelateerde) informatie toevoegt aan de NEFIS bestanden).

Datatypes

De uitwisseling van administratieve gegevens, geometrie, cartografie, meetreeksen, tijdreeksen, oppervlaktewater- en grondwatermodelgegevens moet mogelijk zijn.

Stekkerdoos:

1. Conform de opdracht is de stekkerdoos gebaseerd op GW'96 aangevuld met waardereeksen (tijdreeksen en meetreeksen). Voor geometrie en cartografie zijn geen extra voorzieningen getroffen. Voor uitbreiding van de stekkerdoos met geometrie en cartografie kan worden overwogen om deze gegevenssoorten in GW'96 en/of in de stuurbestanden op te nemen. Zie ook hoofdstuk 3.2.

Documentatie

Alle routines moeten zodanig beschreven zijn dat hun werking duidelijk is. De specificaties voor de call-interfaces moeten volledig zijn.

Stekkerdoos:

1. De functiebeschrijvingen zijn in hoofdstuk 7 opgenomen. In dit hoofdstuk is tevens de samenhang van de functies aan de hand een voorbeeld-programmastuctuur beschreven.

Versienummering

Het versienummer van het bij het inpakken gebruikte classificatiestelsel zoals bepaald door de beheersorganisatie ervan, moet worden opgenomen in het uitwisselingsbestand.

Stekkerdoos:

1. 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 GW'96 versienummer.
2. De hierboven genoemde versie gegevens worden ook naar het uitwisselingsbestand geschreven. Deze gegevens zijn opvraagbaar met een bibliotheekfunctie.

Administratieve relaties
Alle binnen het classificatiestelsel gedefinieerde relaties (foreign keys) moeten uitgewisseld kunnen worden.

Stekkerdoos:

1. De stekkerdoos neemt alle attributen van de entiteittypen mee. Ook biedt de stekkerdoos de mogelijkheid om te achterhalen welke attributen samen de primary key vormen. Adressering in de uitwisselingsfile vindt plaats aan de hand van deze (primaire) sleutels. De stekkerdoos biedt geen faciliteit om te achterhalen of een bepaalde attribuut een foreign key is. Foreign keys worden op dezelfde manier meegenomen als andere attributen en kunnen op dezelfde manier worden gebruikt.

Logging
De werking van het in- en uitpakproces moet worden vastgelegd in logfiles (errors, warnings and results).

Stekkerdoos:

1. Alle door de high-level-schil gesignaleerde fouten en waarschuwingen worden door de high-level-schil geschreven naar een vaste logfile op het werkgebied, voorzien van een datum/tijd indicatie. Bovendien wordt het openen en sluiten van een uitwisselingsbestand naar de logfile geschreven. Een reeds bestaande logfile wordt aangevuld.

Write append
Het moet mogelijk zijn om aan een bestaand uitwisselingsbestand gegevens toe te voegen, ook van objecten die al zijn opgenomen. Ook de volgorde waarin de gegevens worden geschreven moet vrij zijn.

Stekkerdoos:

1. Gegevens die reeds in een bestand zijn opgeslagen kunnen onbeperkt worden gewijzigd. (Dit geldt niet voor de sleutelvelden). Toevoegen is eveneens mogelijk. Verwijderen is niet mogelijk. De te verwijderen velden kunnen als zodanig worden gemarkeerd in de was/wordt status. Zie verder de beschrijving onder was/wordt mutaties.

Read random
Het lezen van de gegevens in een aangeleverd uitwisselingsbestand moet niet aan een bepaalde volgorde gebonden zijn.

Stekkerdoos:

1. Door het gebruik van NEFIS kunnen de gegevens in elke willekeurige volgorde worden gelezen.

Zelfstandig werkende applicatie
De routines moeten door de leverancier zelf worden toegepast in zelfstandig werkende teststekkers. Deze moeten in staat zijn uitwisselingsbestanden te genereren en uit te pakken. uitgaande van een set gegevens van de verschillende typen, in een voorgeschreven bestands-layout.

Stekkerdoos:

1. De teststekker worden gevormd door de stekkers die via afzonderlijke projecten worden gerealiseerd. Voor de tests worden ad-hoc stekkers gebouwd. Dit wordt verder uitgewerkt in het testplan.

Was/wordt
De uitwisseling van was/wordt-mutaties moet worden ondersteund.

Stekkerdoos:

1. Aan alle entiteittypen wordt een was/wordt-status toegevoegd. 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. Voorgesteld wordt om de volgende codering te gebruiken: "N" nieuw, "D" gedeeltelijke herziening, "H" gehele herziening en "V" vervallen.

Onderhoud
Alle informatie die nodig is voor een correcte werking van de routines (waaronder de stuurgegevens) moet gemakkelijk door de eigen organisatie onderhouden kunnen worden. Het classificatiestelsel (zeker het bilaterale) kan wijzigen. Dit moet op een eenvoudige manier doorgevoerd kunnen worden in de routines.

Stekkerdoos:

1. Er wordt een programma beschikbaar gesteld voor het genereren van een stuurbestand vanuit een SDW-model. Wijzigingen in GW'96 vereisen dus geen software aanpassingen.
2. Van NEFIS, de high-level-schil, de stuurfile generator en de NEFIS-definitiefile generator worden zowel de binaire versies als de sources beschikbaar gesteld. Voor de stekkerdoos wordt gebruik gemaakt van de standaard versie van NEFIS. Daarom wordt ook het NEFIS manual beschikbaar gesteld.

Stuurgegevens open
De stuurtabel met het afgesproken classificatiestelsel maakt onderdeel uit van de stekkerdoos. Deze tabel moet in ASCII-formaat toegankelijk zijn. De praktijk zal immers zijn dat uitlevering van de nieuwe tabel enige tijd na de vaststelling van de nieuwe standaard zal plaatsvinden. Wanneer nodig dan kan de tabel zelf worden aangepast aan de nieuwe situatie.

Stekkerdoos:

1. De stuurbestandgenerator genereert een ASCII stuurbestand uit het SDW model. Dit stuurbestand kan met een willekeurige teksteditor worden gewijzigd en aangevuld.

Aanmaak van de stuurgegevens
Het moet mogelijk zijn de classificatie-informatie (stuurgegevens) van de GW'96 automatisch te genereren uit de aangeleverde modelgegevens in SDW.

Stekkerdoos:

1. Dit wordt verzorgd door de stuurbestand generator.

Taal
De routines moeten worden opgeleverd in ANSI C en het eigendomsrecht wordt overgedragen aan de opdrachtgever.

Stekkerdoos:

1. Middels een wijzigingsvoorstel is afgesproken dat de stekkerdoos in Fortran wordt geprogrammeerd, omdat NEFIS ook in Fortran geschreven is. Zou de stekkerdoos in C worden geschreven dan moet in het nadeligste geval eerst van Fortran => C overgegaan worden (grensvlak tussen applicatie en stekkerdoos bibliotheek) en later van C => Fortran (grensvlak tussen high-level-schil en NEFIS). Voor de stekkerbouwer is de stekkerdoos zowel vanuit C, C++ als Fortran te gebruiken..

High level routines

Wanneer de ontwikkelomgeving van de stekkerbouwer dat toestaat, dan moet het mogelijk zijn de routines op een hoog abstractieniveau aan te roepen (bijvoorbeeld: extract_instance (mpnident,21812100001) of extract_column (mpnident,21812100001,mptstik). Hierbij kan dan gebruik worden gemaakt van parameter passing, structures en shared memory. Zo blijven specifieke systeemaangelegenheden (zoals geheugentoewijzing, filehandles en pointers) voor de stekkerbouwer verborgen.

Stekkerdoos:

1. Er wordt een set van hoog niveau bibliotheekfuncties beschikbaar gesteld, waarin de specifieke systeemaangelegenheden maximaal zijn afgeschermd. Voor de uitwerking wordt verwezen naar hoofdstuk 7 met de beschrijving van de systeemfuncties.
2. Het gebruik van shared memory heeft met name zin als snel gegevens van het ene programma naar het andere moeten worden overgedragen, zoals bijvoorbeeld in situaties waarin met een vaste ritme programma's beurtelings moeten draaien. Omdat de stekkerdoos met name gericht is op de uitwisselingsmogelijkheden voor bestanden biedt een shared memory versie op dit moment nauwelijks tot geen meerwaarde. Conform de offerte wordt nu geen shared-memory versie geleverd. Indien gewenst kan de shared-memory optie later alsnog toegevoegd worden.
3. Het realiseren van een gegevensuitwisseling via shared memory kan door een soort van Nefis databestand in het computergeheugen op te slaan. Dit is technisch vrij eenvoudig realiseerbaar. Het aspect waar vervolgens rekening mee moet worden gehouden is dat het geheugengebruik van de verschillende programma's op elkaar moeten worden afgestemd. Dit vereist een platform afhankelijke invulling. Een ander aspect is de synchronisatie tussen de verschillende programma's. Immers, het lezende programma moet veelal wachten tot het schrijvende programma de nodige informatie heeft geschreven. De eenvoudigste oplossing hiervoor is het bouwen van een overkoepelend programma die beurtelings de beide andere programma's start. (Het realiseren van een dergelijke oplossing voor een 32-bits windowsomgeving kost ca 5 dagen.)

Low level routines

Het gebruik van de routines moet ook in Fortran ontwikkelomgevingen geen extra programmeerinspanning vergen. Wanneer dit noodzakelijk is dan moeten hiervoor de interne stekkerdoosroutines rechtstreeks aangeroepen kunnen worden.

Stekkerdoos:

1. Er worden zogenaamde headerfiles met declaraties beschikbaar gesteld welke door de stekkerbouwer kan worden gebruikt. Verder zijn er geen bijzonderheden op dit punt.
2. Het gebruik van NEFIS routines buiten de "high-level-schil" om kan de consistentie van de "high-level-schil" in gevaar brengen. Voor de juiste werking van de "high-level-schil" kan dan niet worden ingestaan. Daarom wordt het gebruik van NEFIS buiten de high-level-schil om sterk afgeraden.

Platforms

De routines moeten bruikbaar zijn op de volgende computerplatforms: Microsoft Windows 32 bits, HP Unix, Digital Unix, Vax VMS, IBM AS/400, IBM RS/6000

Stekkerdoos:

1. Het systeem wordt opgeleverd op een 32-bits Windows omgeving. Conversies naar andere platforms vallen buiten het kader van dit project.

Interne consistentie
In het uitwisselingsbestand moet een vorm van CRC-controle zitten om de kans op fouten bij de uitwisseling te verkleinen.

Stekkerdoos:

1. De Nefis-definitiebestand generator neemt in het Nefis-definitiebestand een unieke code op. Deze code wordt automatisch overgenomen in het uitwisselingsbestand. Als nu een uitwisselingsbestand wordt geopend met een definitiebestand die een andere code bevat, dan wordt een waarschuwing gegeven als indicatie dat een ander Nefis-definitiebestand wordt gebruikt dan het bestand waarmee de uitwisselingsfile is gegenereerd. Het is aan de stekkerbouwer om te beoordelen in hoeverre dit toelaatbaar is.
2. Verder zal een checksum aan de beide Nefis-bestanden worden toegevoegd.

Administratieve datatypen
Alle datatypen die in de GW'96 voorkomen (alfanumeriek, numeriek, datum en logical) moeten door de stekkerdoos verwerkt kunnen worden. Veldlengtes moeten onbeperkt zijn.

Stekkerdoos:

1. In GW'96 komen geen Logical velden voor. Daarom worden in de stekkerdoos geen faciliteiten voor logical velden opgenomen.
2. In GW'96 zijn de lengtes van de velden aangegeven. De stekkerdoos neemt deze lengtes over in het uitwisselingsbestand.

3.2 Additionele uitgangspunten/randvoorwaarden

Afstemming van applicaties op elkaar blijft nodig
Middels GW'96 is een classificatie vastgelegd van de relevante gegevens. De naamgeving (of identificatie) van de in de natuur voorkomende objecten is echter niet vastgelegd. Daardoor kan het voorkomen dat voor één en hetzelfde object twee namen (of identificaties) worden gebruikt. Bij het uitwisselen van gegevens blijft daarom onderlinge afstemming noodzakelijk. Dit is zeer noodzakelijk als gegevens uit de uitwisselingsfile worden gebruikt in combinatie met reeds aanwezige gegevens. In dat geval kan het voorkomen dat in de applicatie twee objecten zijn gedefinieerd, terwijl in de natuur slechts één object aanwezig is.

Geometrie
Op functioneel gebied moeten de geometrische objecten nog worden gedefinieerd (of zo men wil geclassificeerd). Dit vraagt additionele inspanning en intensief overleg met deskundigen. Bij deze definitie kan gebruik worden gemaakt van de logische structuren van de objecten zoals gedefinieerd in NEN-1878 en NEN-3610.

Als deze objecten zijn gedefinieerd, dan ligt het voor de hand om een paar generieke functies te bouwen voor het lezen en schrijven van deze objecten.

4 Gewenst systeem

4.1 Globale systeembeschrijving

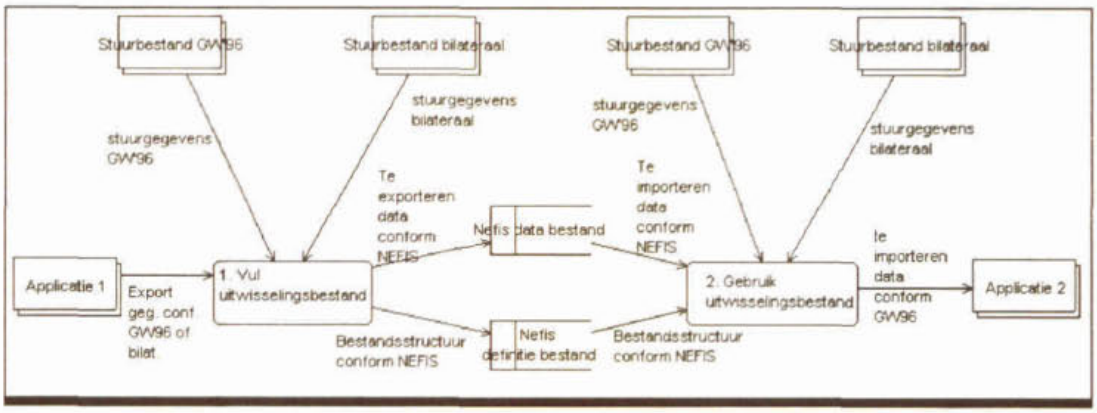
Na een grondige analyse van de gevraagde functionaliteit, in beschouwing nemend de voorgestelde oplossingsrichting in het Functioneel Ontwerp ("Functioneel Ontwerp Stekkerdoos Water"; 30 oktober 1996) en een analyse van de door de opdrachtgever aangegeven mogelijke alternatieve uitwisselingsformaten, is gekozen voor een oplossing op basis van de NEFIS-bibliotheek. Rond deze NEFIS bibliotheek wordt een schil met "high-level" routines gerealiseerd zodat de stekkerbouwers op een eenvoudige manier stekkers kan bouwen.

4.2 Systeem context

Deze paragraaf beschrijft het functioneren van de stekkerdoos in zijn omgeving (context). De eerste stap voor het gebruik van de stekkerdoos is het klaarzetten van het GW'96 stuurbestand en het eventueel noodzakelijke bilaterale stuurbestand. Het klaarzetten van de stuurbestanden is de taak van de stekkerbouwer. Voor het aanmaken van een GW'96 stuurbestand wordt een programma beschikbaar gesteld dat vanuit het GW'96 SDW-model een stuurbestand genereert. Het stuurbestand met bilaterale afspraken moet met de hand worden gemaakt met dezelfde structuur/formaat als het GW'96 stuurbestand. Gegevenselementen of entiteittypen die in GW'96 voorkomen kunnen worden aangepast door in het bilaterale stuurbestand gewijzigde definities op te nemen. Deze beide stuurbestanden worden door een hulpprogramma omgezet in een NEFIS-datastructuur, welke door de stekkerdoos worden gebruikt bij het schrijven en lezen van de NEFIS-databestanden. Tevens genereert dit hulpprogramma headerfiles met declaraties die het de stekkerbouwer eenvoudig(er) maken om stekkers te bouwen.

De stekkerbouwer bouwt een stekker met gebruik van de functies van de stekkerdoos. Deze stekker kan een apart conversieprogramma zijn, maar ook een integraal onderdeel van een applicatie. De stekker schrijft naar een NEFIS-databestand die de structuur heeft zoals beschreven in het NEFIS-definitiebestand. Daarom moet bij uitwisseling altijd zowel het data-bestand als het definitiebestand worden uitgewisseld. De stekkerbouwer die de bestanden wil lezen hoeft niet te beschikken over de stuurbestanden, omdat alle informatie beschikbaar is, en opvraagbaar is vanuit het NEFIS-definitiebestand. In dat geval is het echter aan te bevelen om ook de door de Nefis-structuurgenerator gegenereerde headerfiles met declaraties mee te zenden, omdat dit het bouwen van een "lees-stekker" vereenvoudigt. (In de toekomst kan een optie worden ontwikkeld voor het genereren van de headerfiles met declaraties op basis van het NEFIS-definitie- en -databestand. Deze functie wordt nu nog niet ontwikkeld omdat het uitgangspunt is dat elke organisatie de beschikking heeft of krijgt over de stuurbestanden, en dus zelf de include-files kan genereren)

Een en ander is grafisch weergegeven in figuur 2.

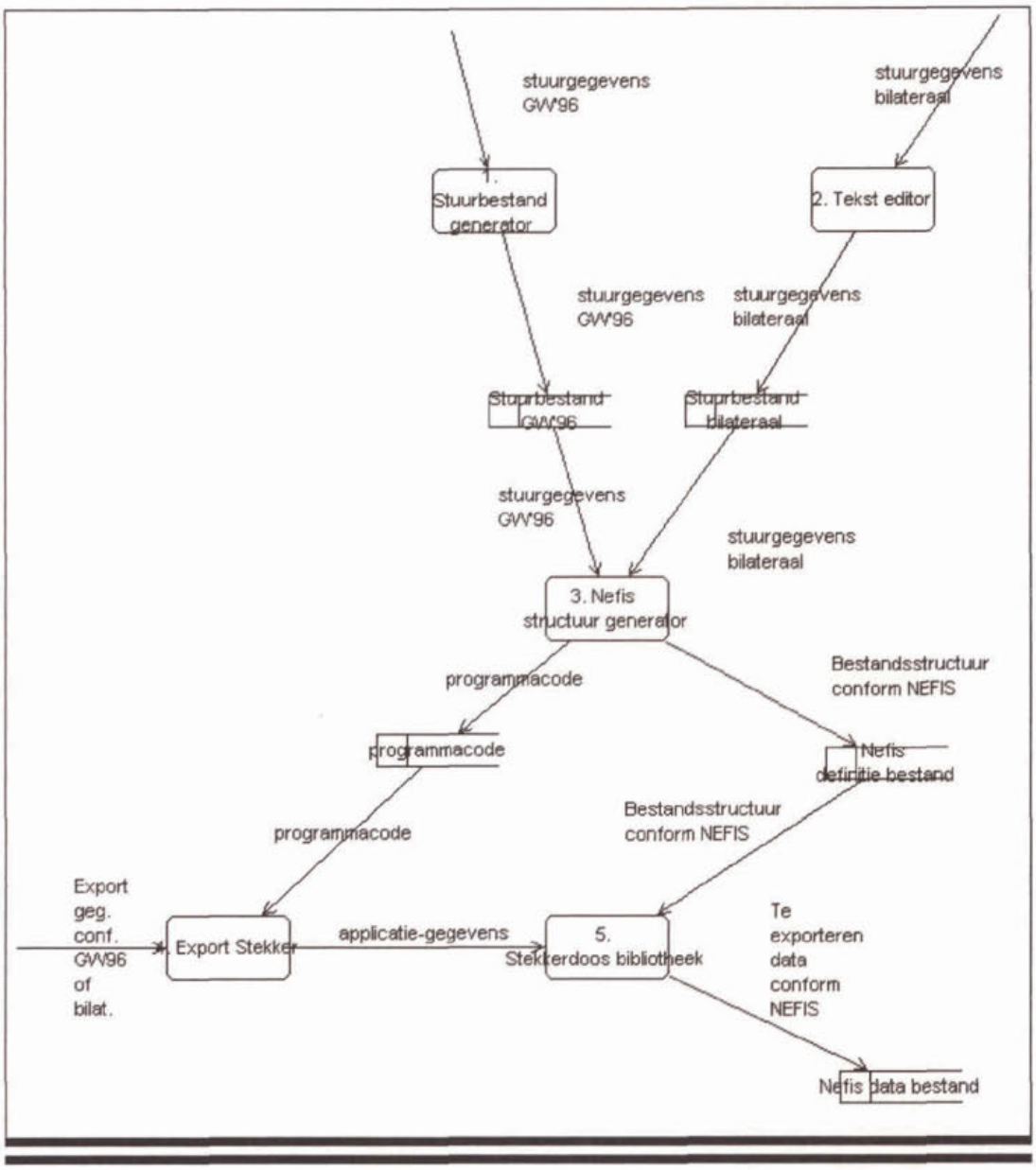


Figuur 1 Context diagram

5 Functioneel ontwerp

5.1 Functioneel model

Figuur 3 geeft in hoofdlijnen de functionaliteit van de module Vul-uitwisselingsbestand. De module Gebruik-uitwisselingsbestand is analoog.



Figuur 2 Gegevensuitwisseling

De **stuurbestandgenerator** genereert het GW-stuurbestand vanuit GW-96. Dit programma voegt aan alle entiteittypen tevens een was/wordt status (als attribuut) toe. Parallel aan het genereren van het GW-stuurbestand kan met de hand (teksteditor) een stuurbestand gemaakt worden met bilaterale afspraken. Deze beide bestanden worden daarna door de NEFIS-structuurgenerator omgezet in een NEFIS-definitiebestand. Tegelijk worden headerfiles met declaraties gegenereerd met declaraties voor Fortran en C stekkerprogramma's.

De **exportstekker** biedt de applicatiegegevens via de stekkerdoosbibliotheek (high-level-schil) aan NEFIS aan. NEFIS zorgt dan voor de verdere opslag en uitwisseling. Deze stekkers zijn applicatieafhankelijk en kunnen pas worden gebouwd als duidelijk is welke informatie in welke vorm met de applicatie moet worden gecommuniceerd.

De **stekkerdoos bibliotheek** is een set van functies die vanuit Fortran en C kunnen worden aangeroepen en die het lezen/schrijven naar het NEFIS data bestand verzorgen. Met deze functies kan de stekkerbouwer op een eenvoudige manier conversieprogramma's (stekkers) maken naar het uitwisselingsformaat en omgekeerd. Deze bibliotheek is de "high-level-schil" rond NEFIS. In deze schil wordt een selectie gemaakt uit de mogelijkheden van NEFIS die op een toegankelijke manier aan de stekkerbouwer worden aangeboden. In deze schil worden bovendien functies zoals logging en foutafhandeling gerealiseerd. Voor de stekkerdoos wordt gebruik gemaakt van versie 3.01 van NEFIS.

In de stekkerdoosbibliotheek is sprake de volgende categorien functies:

- het openen en sluiten van een uitwisselingsbestand
- het aanmaken van GW'96 entiteittypen en het lezen en schrijven van GW'96 attributen
- het lezen/schrijven van waardereeksen
- opvraagfuncties voor het opvragen van bestandsinformatie, informatie van entiteittypen en voor het opvragen van informatie van een attributen.

De lees en schrijffuncties van GW'96 attributen zijn weer in twee categorieen te verdelen:

- het lezen van een attribuut op basis van sleutelinformatie
- het lezen van een attribuut afhankelijk van de actuele postitie in het bestand (met in achtneming van de sleutelinformatie).

Voor het lezen en schrijven van waardereeksen zijn speciale functies gedefinieerd:

- het lezen en schrijven van 1 dimensionale gegevens
- het lezen en schrijven van meer dimensionale gegeven.

Dit onderscheid tussen 1- en meer-dimensionale waardereeksen is gemaakt omdat de 1-dimensionale functies gemakkelijker zijn in het gebruik dan de n-dimensionale functies en de verwachting is dat de 1-dimensionale functies veel gebruikt zullen worden.

De GW'96 lees en schrijf routines kunnen niet worden gebruikt voor het lezen en schrijven van waardereeksen en omgekeerd kunnen de waardereeks-routines niet worden gebruikt voor het lezen van GW'96 attributen en entiteittypen. GW'96 gegevens worden geadresseerd via sleutelwaarden en waardereeksinformatie wordt geadresseerd via indices.

De waardereeksen kunnen in de stuurbestanden worden gedefinieerd door achter de UvW code van de betreffende entiteittype de dimensies aan te geven, waarbij altijd één dimensie (de laatste) wordt opgegeven met een sterretje "*" als indicatie dat de grootte van deze dimensie variabel is en pas "in run-time" wordt vastgesteld. Ook voor 1-dimensionale waardereeksen moet een "*" worden toegevoegd als teken dat het een waardereeks betreft.

Uit het stuurbestand is dus af te leiden welke entiteittypen "waardereeksen" zijn en welke standaard-entiteittypen zijn.

De stekkerdoos bibliotheek is in hoofdstuk 7 functioneel beschreven.

5.2 Gegevensmodel

In de stekkerdoos zijn drie soorten van bestanden:

- NEFIS bestanden
- Headerfiles met declaraties
- Stuurbestanden.

De structuur van de stuurbestanden is beschreven in de gebruikershandleiding hoofdstuk 6.

De **NEFIS-definitiebestanden** zijn beschreven in de NEFIS handleiding. Uitgebreide kennis over de structuur van deze bestanden vergroten het inzicht in de gebruiksmogelijkheden van NEFIS niet. Voor de NEFIS-definitiebestanden gelden de volgende additionele afspraken:

Omdat de UvW-codes van de entiteittypen maximaal drie characters bevatten, en NEFIS 16 posities kan adresseren, wordt de positie 16 gebruikt voor een aanduiding welke aangeeft of het een entiteittype betreft uit het bestand met de bilaterale afspraken of uit de GW-classificaties. (code B is bilateraal; anders GW'96)
Voor attributen wordt dezelfde werkwijze gehanteerd..
Bij de beschrijving van entiteittypen (NEFIS-cellen genoemd) wordt bij de attributen op analoge wijze aangegeven op positie 15 of het een sleutelattribuut betreft of niet.
Er worden een aantal extra gegevenselementen in het NEFIS-definitiebestand gedefinieerd voor:

- de versie van de GW'96 classificatie,
- de naam van het SDW-model,
- de datum waarop het GW'96 stuurbestand is gegenereerd,
- de versie van het bilaterale stuurbestand,
- de toepassing (de reden waarom het bilaterale stuurbestand is aangemaakt)
- de datum waarop het bilaterale stuurbestand is aangemaakt,
- de identificatie code om na te gaan of het definitiebestand en het databestand bij elkaar horen.

De bovenstaande afspraken met betrekking tot NEFIS worden geheel binnen de stekkerdoosfuncties afgehandeld en zijn voor de stekkerbouwer afgeschermd.

De structuur van de **NEFIS-databestanden** is in hoofdstuk 9 in hoofdlijnen beschreven.

In de Nefis-databestanden zijn twee categorieen van gegevens te onderscheiden:

- gegevens die vallen binnen de classificatie van GW'96
- n-dimensionale waardereeksen

Het onderscheid tussen deze gegevens wordt in het stuurbestand aangegeven door voor waardereeksen de dimensies aan te geven, waarbij ééndimensie (de laatste) wordt aangegeven met een "*" als teken dat deze dynamisch is. Bij een 1-dimensionaal entiteittype kan dus worden volstaan met een "*". Voor GW'96 gegevens wordt geen "*" opgegeven. Het maximum aantal dimensies is 5. Een meer uitgebreide beschrijving van de functionaliteit voor waardereeksen is opgenomen in hoofdstuk 5.3

De **headerfiles met declaraties** (ook wel "include-files" genoemd) voor Fortran en C bevatten declaraties van alle gegevenselementen. Bovendien zijn een aantal parameters in deze files opgenomen, zoals het maximum aantal sleutels dat kan worden meegegeven en dergelijke.

De stuurbestanden zijn in detail beschreven in hoofdstuk 8:

5.3 Waardereeksen

5.3.1 Algemeen

De stekkerdoos kan naast de GW'96 gegevens ook waardereeksen lezen en schrijven. Het verschil tussen waardereeksen en GW'96 gegevens is dat waardereeksen veelal lange reeksen van gegevens zijn met een vaste volgorde. Deze gegevens voldoen niet aan het relationele model omdat de waarden niet via sleutels benaderd kunnen worden en er kunnen dus geen relaties worden gelegd.

Relationeel model

stuw-identificatie	hoogte	breedte
stuw-1	40	60
stuw-2	50	80
...	....	....

adressering met behulp van een sleutel

Eén dimensionale waardereeks gemeten op 9 januari 1997 van 0:00 tot 12:00 (equidistant met een interval van 10 minuten)

meetwaarde-1	meetwaarde-2
40	60
50	80
20	30
....	.....

adressering met behulp van een volgnummer

Het is voor 1-dimensionale waardereeksen mogelijk om deze op te slaan in een relationeel model door aan alle metingen een sleutelwaarde toe te voegen. Voor 2-dimensionale waardereeksen wordt dit al een stuk moeilijker en voor 3 of meer dimensionale waardereeksen moet veel administratie worden opgenomen om de 3-dimensionale structuur vast te leggen.

5.3.2 Waardereeksen in de stekkerdoos

In de stekkerdoos worden waardereeksen beschouwd als een meer-dimensionale structuur van gelijkvormige sets van gegevens zoals bijvoorbeeld een aantal metingen op 1 punt op 1 tijdstip. De structuur van de sets kan in een stuurbestand worden vastgelegd, door hiervoor een entiteittype te definieren. Met uitzondering van één, moeten voor de waardereeks-entiteittypen ook de dimensies en hun groottes worden opgegeven in het stuurbestand. De grootte van die ene dimensie wordt aangegeven met een "*" teken.

De Nefis-structuurgenerator creëert het entiteittype welke vanuit de stekker kan worden gebruikt.

In de stekker kan met behulp van de routine MAKWRD in "run-time" een waardereeks worden gecreëerd met een naam die in "run-time" wordt bepaald met gebruik van de definitie vanuit het stuurbestand. De maximale lengte van de naam van de waardereeks is 14 posities

Op deze wijze is mogelijk om waardereeksen te creëren waarnaar wordt verwezen vanuit andere entiteittypen.

Bijvoorbeeld:

Metrische kenmerken-objectinformatie

identificatie	coördinaten stelsel	id puntenreeks	kleur	.....
stuw-1	RD	**stuw1rks**		
stuw-2	RD	**stuw2rks**		

puntenreeks: **stuw1rks**

interpolatie	x	y	z

puntenreeks: **stuw2rks**

interpolatie	x	y	z

Op heb moment dat een waardereeks wordt aangemaakt wordt tegelijk de naam van de waardereeks geschreven in de entiteit/attribuut van waaruit naar de waardereeks wordt gerefereerd. De plaats van deze referentie wordt tevens opgeslagen bij de waardereeks, zodat deze plaats voor elke waardereeks achterhaald kan worden. De Stekkerdoos Water bevat geen "integriteitscontroles" met betrekking tot deze referenties. Dit houdt in dat als een verwijzing door een andere waarde wordt overschreven, dat de stekkerdoos dit niet automatisch signaleert.

5.4 Externe interfaces

De afbakening van het systeem in zijn omgeving is in voorgaande hoofdstukken en paragrafen beschreven. Verder zijn er geen externe interfaces.

5.5 Gebruikersinteractie

De programma's stuurbestandgenerator en NEFIS-structuur generator zijn 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.

De stekkerdoosbibliotheek bevat alleen maar functies die vanuit Fortran, C en C++ kunnen worden aangeroepen. Deze functies stellen geen vragen aan gebruikers.

5.6 Technische eisen/ontwerp-beslissingen

1. In het SDW model van GW'96 zijn entiteittypen geclusterd in superentiteittypen. Deze superentiteittypen hebben attributen die voor alle subentiteittypen van toepassing zijn. In de stuurtabel 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.
2. Momenteel zijn in GW'96 alle sleutelvelden alfanumeriek. Hier wordt in de stekkerdoos mee gewerkt. Dit heeft als consequentie dat als in de toekomst met numerieke sleutelvelden

wordt gewerkt deze eerst moeten worden geconverteerd naar alfanumeriek voordat deze aan de stekkerdoos kunnen worden aangeboden.

3. Het bestand met de bilaterale afspraken heeft altijd prioriteit boven GW'96.
4. De NEFIS routines kunnen in principe ook door de stekkerbouwer worden gebruikt, maar dit wordt afgeraden omdat daardoor de bewaking verloren gaat die binnen de "high-level-schil" wordt verzorgd. Daarom kan bij rechtstreeks gebruik van NEFIS de correcte werking van de schil niet worden gegarandeerd.
5. De bibliotheek wordt geschreven in Fortran-77 met een C-interface.
6. Het is niet mogelijk om entiteittypen door middel van C-structures door te geven aan de stekkerdoos, omdat de gegevens dan niet weer door een Fortran programma kunnen worden gelezen.
7. Er worden geen speciale voorzieningen getroffen voor het gebruik in Fortran-90 applicaties.
8. De stekkerdoos is ongevoelig voor verschillen tussen hoofdletters en kleine letters voor UvW-codes.
9. De Stuurbestand generator wordt geprogrammeerd met behulp van SDW scripts en kan alleen worden gedraaid als SDWrite is geïnstalleerd.
10. De in dit ontwerp genoemde Teksteditor is een standaard editor waarmee ASCII-bestanden kunnen worden aangemaakt. Deze editor is geen onderdeel van dit levering.
11. Niet gevulde velden worden geinitialiseerd: Integers en reals op de maximaal mogelijke waarden en voor characterstrings wordt op de eerste positie een onleesbaar character neergezet.
12. Als geen type (alfanumeriek, numeriek of datum) is opgegeven in het stuurbestand dan wordt het veld verondersteld een alfanumeriek veld te zijn.
13. Als geen lengte is opgegeven in het stuurbestand dan wordt de lengte 8.0 verondersteld.
14. Als in het stuurbestand een sleutelveld langer is dan 24 posities, dan wordt deze afgekapt op 24 posities. Een sleutelveld dat korter is dan 24 posities wordt intern behandeld alsof deze 24 posities had.
15. Datum velden in het stuurbestand zonder lengte worden verondersteld 8 posities lang te zijn
16. Bij alfanumerieke velden in het stuurbestand worden eventueel opgegeven decimalen achter de punt genegeerd.
17. Het include statement is geen ANSI Fortran. Toch wordt dit statement gebruikt om de onderhoudbaarheid van de code te vergroten. Op bijna alle platforms is wel een equivalente functie te vinden.
18. Entiteiten met sleutelattributen die niet alfanumeriek zijn worden genegeerd.
19. Waardereeksen worden niet geinitialiseerd.
20. Bij gelijke UvW codes in het stuurbestand wordt alleen de eerst voorkomende opgenomen. De latere definities worden na een foutmelding genegeerd.

6 Organisatorische aspecten

6.1 Betrokkenheid van de gebruikers bij projectontwikkeling

De gebruikers worden vertegenwoordigd in de Technische commissie/Begeleidingscommissie en hebben hiermee een sturende invloed op de ontwikkeling. Voor geometrie en cartografie is inbreng van de gebruikers nadrukkelijk vereist om te komen tot goede concepten (dit is echter voorlopig nog buiten het blikveld van deze opdracht).

6.2 Afstemming met stekkerbouwers

De "semi-definitieve" specificaties van de stekkers kunnen pas na goedkeuring van het functioneel ontwerp aan de stekkerbouwers worden gegeven. Een goede en constructieve samenwerking en afstemming tussen het stekkerdoosproject en de stekkerbouwers is daarom noodzakelijk.

6.3 Testen

Hiervoor wordt een apart document opgesteld.

6.4 Beheer en onderhoud

Tijdens de garantie periode wordt een voorstel gedaan voor het beheer en onderhoud van de stekkerdoos. Het beheer en onderhoud van de stekkers valt vooralsnog buiten het kader van dit project.

7 Beschrijving stekkerdoos bibliotheek

Dit hoofdstuk geeft een beschrijving van de stekkerdoosfuncties. In de eerstvolgende paragraaf wordt een samenvatting gegeven van de functies en in de daaropvolgende wordt een beschrijving gegeven van de afzonderlijke functies.

Er is in feite sprake van de volgende categorieen functies:

- openen en sluiten van een uitwisselingsbestand (OPNUWB en CLSUWB)
- creëren van entiteiten en waardereeksen in het databestand (MAKENT en MAKWRD)
- lezen en schrijven van attributen behorend bij entityttypen (RD_KEY , WR_KEY, WRCRNT, RDCRNT en NXTOBJ)
- functies voor het lezen en schrijven van n-dimensionale waardereeksen. (WR1DWR, WRNDWR, RD1DWR en RDNDWR)
- opvraagfuncties het opvragen van:
  - bestandsinformatie (INQUWB),
  - alle in het uitwisselingsbestand aanwezige entiteiten en waardereeksen (INQCNT),
  - informatie van een entiteittype (INQENT) of een waardereeks (INQWRD) en
  - van informatie van een attribuut (INQATT)

De functies WRCRNT en RDCRNT zijn ontworpen om te vermijden dat als een reeks van attributen van één record binnen één entiteittype worden gevraagd of geschreven, dat te veel rekentijd verloren zou gaan met het zoeken en vergelijken van sleutelwaarden. In feite is hier sprake van een optimalisatie.

De functie NXTOBJ is ontworpen om de stekkerbouwer de mogelijkheid te bieden om sequentieel alle aanwezige records van één entiteittype te doorlopen zonder dat de sleutelwaarden bij hem bekend behoeven zijn.

De functies zijn in de Gebruikershandleiding in detail beschreven. Globaal zijn de volgende functies aanwezig:

Algemeen

OPNUWB: Open uitwisselingsbestand

CLSUWB: Sluit het data bestand

GW'96 uitwisseling

MAKENT: Creëer een entiteittype

WR_KEY: Schrijf een attribuutwaarde met sleutel

WRCRNT: Schrijf nog een attribuutwaarde naar het laatst geselecteerde record

RD_KEY: Lees een waarde van het eerste record dat overeenkomt met de opgegeven sleutelwaarden

RDCRNT: Lees nog een waarde van het laatst geselecteerde record

NXTOBJ: Selecteer het volgende record dat voldoet aan de sleutelwaarden

Uitwisseling van waardereeksen

MAKWRD: Creëer een waardereeks met de naam "waardereeksnaam" en structuur "entiteittype"

WR1DWR: Schrijf (een gedeelte van) een 1 dimensionale waardereeks

WRNDWR: Schrijf (een gedeelte van) een n dimensionale waardereeks (n <= 5)

RD1DWR: Lees (een gedeelte van) een 1 dimensionale waardereeks

RDNDWR: Lees (een gedeelte van) een n dimensionale waardereeks (n <= 5)

Informatie opvraagfuncties

INQUWB: Vraag bestandsinformatie

INQCNT: Vraag bestandsinhoud

INQENT: Vraag entiteittype informatie

INQWRD: Vraag waardereeksinformatie

INQATT: Vraag attribuut informatie

De declaraties van de GW'96 attributen zijn in include-/header-files beschikbaar. De namen van de variabelen zijn in deze includefiles gelijk aan de namen van de UvW-codes van de betreffende gegevenselementen. In ANSI fortran mogen variabelen maximaal 6 posities lang zijn. Omdat de UvW-codes meer dan 6 characters lang zijn wordt aanbevolen om bij de bouw van de stekkers op dit punt van de ANSI standaard af te wijken. Indien de stekker strikt moet voldoen aan de ANSI standaard kunnen de standaard meegeleverde declaraties niet worden gebruikt

Het gebruik van de routines WRCRNT en RDCRNT zijn ontworpen om de gebruiker de mogelijkheid te bieden om snel een groot aantal waarden in één record weg te schrijven.

De Gebruikershandleiding bevat ook enkele voorbeeldprogramma's voor Fortran en C.

8 Beschrijving van de stuurtabellen

Dit hoofdstuk is verplaatst naar de Gebruikershandleiding.

9 Beschrijving van Nefis

Ontstaan
NEFIS is ontstaan na een langdurig traject van vooronderzoek en oordeelsvorming om te komen tot een universeel mechanisme voor de opslag en overdracht van gegevens van de WL-modellen.
In dit onderzoek zijn ook de documenten over de toenmalige versie van NETCDF (dit pakket heette toen nog CDF) bestudeerd. Afgezien van een aantal toenmalige technische bezwaren (zoals de platformafhankelijkheid van de datafiles, de onmogelijkheid om meerdere NETCDF-variabelen in één file op te slaan) was het belangrijkste knelpunt het beschermingsbeleid van de Amerikaanse Ministerie welke verspreiding buiten Amerika verbood. WL heeft toen besloten om een eigen pakket te ontwikkelen, waarin enerzijds de inmiddels opgedane ervaringen met relationele databases, en anderzijds de ervaringen met abstracte datastructuren werden overgenomen.

Structuur van de NEFIS datafiles
De basis van een NEFIS file zijn de gegevensstructuren zoals deze zijn opgenomen in programmeertalen als C en Pascal, waarin het mogelijk is om middels één naam een setje van variabelen te adresseren. Dit noemen wij een "C-structure" en in NEFIS heet dit een "Cel". Een Cel kan er nu bijvoorbeeld als volgt uit zien:

Naam van de cel

Integer*(24)	naam1 (dit is een array met 24 integer waarden)
Integer	naam2 (dit is een enkelvoudig element)
Character*50	naam3 (dit is een tekststring van 50 posities)
.....	

In deze cel zitten dus gegevenselementen. Aan gegevenselementen kunnen beschrijvingen worden toegevoegd zoals o.a. de gebruikte eenheid en de betekenis van het gegevenselement. De gegevenselementen kunnen in verschillende Cellen voorkomen. (Vergelijking met een SDW gegevensmodel levert de volgende overeenkomsten: een gegevenselement heet in SDW ook een gegevens element en een cel is te vergelijken met de definitie van een entiteittype.)

In een relationele database kunnen voor één entiteittype meerdere records voorkomen, zo ook in NEFIS. In NEFIS kunnen zelfs meerdimensionale (maximaal 5 dimensies) reeksen met cellen worden gedefinieerd, waarbij één dimensie onbeperkt kan oplopen. In NEFIS termen heet een dergelijke reeks een "**Groep**".

In een NEFIS datafile kunnen een onbeperkt aantal groepen worden opgenomen, die onafhankelijk van elkaar kunnen groeien (Dit in tegenstelling tot NETCDF waar de groei van de ene "groep" gekoppeld is aan de groei van de andere groepen in dezelfde datafile.

Per groep kunnen bovendien kenmerken ("Attributen") worden toegevoegd (evenals in NETCDF).

Vergelijking met relationele databases
NEFIS is primair ontwikkeld voor de uitwisseling van meerdimensionale waardereeksen, en tevens geschikt voor de uitwisseling van de bijbehorende administratieve gegevens en de metagegevens. Bij het WL zijn NEFIS-files in gebruik van enkele honderden Megabytes, die met een zeer goede performance gelezen en geschreven kunnen worden.

Relationele databases zijn ontwikkeld voor het opslaan en beheren van met name administratieve gegevens, en bieden daarvoor veel faciliteiten en een optimale performance.

Het is mogelijk om waardereeksen op te slaan in een relationele database door voor elke waarde een record te creëren. Het opslaan van 2 dimensionale reeksen is problematischer en 3 dimensionale reeksen is nagenoeg onmogelijk. Dit is niet verwonderlijk gezien het feit dat een relationele database hiervoor niet is ontworpen. In NEFIS kunnen zelfs 5 dimensionale reeksen worden opgeslagen, door een 5 dimensionale groep aan te maken.

Geometrische gegevens
In NEFIS kunnen geometrische gegevens worden opgeslagen door voor alle geometrische primitieven een cel te definiëren. Daarna kan in andere groepen onbeperkt worden verwezen naar deze geometrische primitieven en daar de specifieke gegevens aan toevoegen.

Bijvoorbeeld:
Voor de primitieve "lijnstuk" wordt een cel met de naam "lijnstuk" gedefinieerd, die bestaat uit een 3-dimensionale richtingsvector en een lengte.
Voor de definitie van een lijn die is samengesteld uit een groot aantal lijnstukken kan een groep worden gedefinieerd bestaande uit een 1 dimensionale reeks van "lijnstukken". Het beginpunt kan worden vastgelegd in een bij de groep behorende attribuut. Om een driedimensionaal onregelmatig rooster te definiëren kan worden volstaan met het definiëren van een 3-dimensionale groep met "lijnstukken".

Dan volgt nu een voorbeeld van de filestructuur voor het opslaan van een 2-dimensionaal onregelmatig rooster en een serie Naam-Adres-Woonplaats gegevens:

Cel: Lijnstuk

Integer*4	X_Y_Z_vector(3)
Integer*4	Lengte

Groep 2D-rooster Attribuut= "Beschrijving"
Inhoud:

lijnstuk(1,1)	lijnstuk(1,2)	lijnstuk(1,3)	lijnstuk(1,4)
lijnstuk(2,1)	lijnstuk(2,2)	lijnstuk(2,3)	lijnstuk(2,4)
lijnstuk(3,1)	lijnstuk(3,2)	lijnstuk(3,3)	lijnstuk(3,4)
lijnstuk(4,1)	lijnstuk(4,2)	lijnstuk(4,3)	lijnstuk(4,4)
lijnstuk(5,1)	lijnstuk(5,2)	lijnstuk(5,3)	lijnstuk(5,4)
lijnstuk(6,1)	lijnstuk(6,2)	lijnstuk(6,3)	lijnstuk(6,4)
.....	.....	.....	.....

Cel: NAW

Char*36	Naam
Char*36	Straat
Integer*4	Nr
Char*8	Postcode
Char*36	Woonplaats
.....	.....