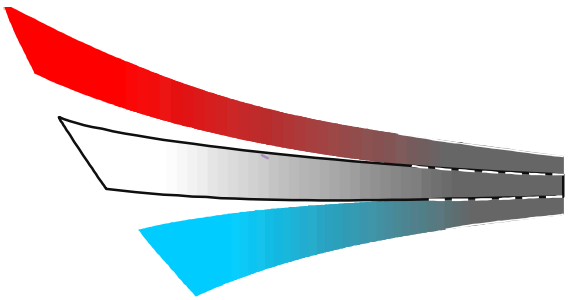

IVERA

Functionele specificatie

Versie 3.01

7 juli 2014

Een uitgave van
Stichting Beheer IVERA protocol
Zoetermeer, Nederland

Pub. No.: IVERA FS 3.01

Datum: 7 juli 2014

Titel: IVERA Functionele specificatie (versie 3.01)

Mocht u fouten of onvolledigheden ontdekken, of suggesties voor verbetering hebben, dan stellen wij het zeer op prijs, als u deze stuurt naar:

Stichting Beheer IVERA protocol
Postbus 190
2700 AD Zoetermeer

Voorwoord

Nederland kent een groot aantal geregelde kruispunten voorzien van verkeersregelinstallaties. De verkeersregelinstallaties zijn in beheer bij Rijkswaterstaat, provincies en gemeentes. Voor een adequaat beheer van de verkeersregelinstallatie is uniformiteit in beheer een noodzaak, vooral voor beheerders met verkeersregelinstallaties van verschillende fabrikanten in hun park.

Het IVER en ASTRIN hebben de noodzaak tot standaardisatie onderkend en hebben de wens uitgesproken in de toekomst alle nieuwe verkeersregelinstallatie te voorzien van een gestandaardiseerde communicatie-interface voor de communicatie met een beheerscentrale.

Enkele jaren geleden leidde dit tot het verschijnen van een eerste versie -1.30- van het IVERA-protocol. Inmiddels is deze versie van het protocol in gebruik in meer dan 500 verkeersregelinstallaties.

Nu er enige tijd ervaring is opgedaan met het gebruik van dit protocol, wordt de behoefte gevoeld bij wegbeheerders en fabrikanten het protocol te verbeteren en uit te breiden op basis van die ervaring.

Dit heeft geleid tot het door de Technische Werkgroep van de Stichting Beheer IVERA-protocol opstellen van een specificatie voor versie 3.01 van het protocol.

Deze specificatie bestaat uit de volgende documenten:

- Een functionele specificatie van het IVERA-protocol (niet toepassing specifiek).
- Een beschrijving van alle objecten en hun eigenschappen zoals die voorkomen in de toepassing van het IVERA-protocol voor de communicatie tussen een verkeersregelinstallatie en een beheerscentrale.
- Een technische specificatie

Inhoudsopgave

1. Samenvatting

Het IVERA-protocol heeft tot doelstelling een fabrikant onafhankelijke oplossing voor communicatie tussen een beheerscentrale en een VRI. Het gebruik is echter niet beperkt tot deze toepassing en kan ook in andere communicatie systemen gebruikt worden.
De opzet van het IVERA-protocol gaat zoveel mogelijke uit van het gebruik van standaard communicatie faciliteiten voor zowel infrastructuur als communicatie software. Daartoe bevindt het IVERA-protocol zich op laag 7 “applicatie laag” van het OSI-model. Tevens wordt alleen gebruik gemaakt van ASCII tekst in de communicatie. Verder is gekozen voor een master/slave protocol waarin de beheerscentrale master is en de VRI slave.

Functioneel levert het IVERA-protocol de mogelijkheid te lezen van en schrijven naar zogeheten objecten in de slave. De objecten hebben een unieke naam en worden door een instantie beheerd zodat namen niet dubbel uitgegeven kunnen worden. Elk object heeft een aantal attributen die de technische aspecten van een object beschrijven zoals omschrijving, type (soort gegevens), pincode (ter beveiliging van schrijf acties), het aantal elementen et cetera. De data van het object is in de slave opgeslagen en kan benaderd worden via de object definitie. Op deze wijze kan de beheerscentrale gegevens uit de VRI lezen of gegevens naar de VRI schrijven.

De uiteindelijke functionaliteit van de slave wordt bepaald door de aanwezigheid van objecten. Om deze functionaliteit vast te kunnen stellen zijn standaard objecten gedefinieerd waarmee de beheerscentrale kan vaststellen welke objecten ondersteund worden en wat het type van de slave is.

Ter beveiliging zijn gebruikers groepen gedefinieerd met behulp van pincodes. Het kenbaar maken van een bepaalde pincode geeft het recht objecten gekoppeld aan de betreffende gebruikersgroep of een lagere gebruikersgroep te manipuleren.

2. Inleiding

Dit document bevat een algemene beschrijving van het IVERA-protocol. Het IVERA-protocol is een initiatief van het IVER, CVN en de ASTRIN. De naam IVERA is een samenvoeging van IVER en ASTRIN.

Het IVERA-protocol is in eerste instantie ontwikkeld voor communicatie tussen verkeersregelinstallaties en een beheerscentrale. Naast deze toepassing is het IVERA-protocol ook geschikt voor andere toepassing zoals:

- Communicatie met toeritdoseertoestellen
- Aansturing van borden in een parkeerverwijzingssysteem
- Koppeling tussen beheerscentrales onderling

Bij de opzet van het IVERA-protocol wordt zoveel mogelijk uitgegaan van het gebruik van standaard communicatie faciliteiten voor zowel de communicatie infrastructuur als de communicatiesoftware. De voordelen van deze aanpak zijn:

- Ondersteuning van diverse communicatienetwerken.
- Ondersteuning van gangbare protocollen.
- Fabrikantonafhankelijke oplossing.
- Minimale ontwikkelingsinspanning door het gebruik van standaard hardware en software componenten.

De technische keuzen ten aanzien van communicatie netwerken vallen buiten de scope van dit document. Hiervoor wordt verwezen naar het technisch ontwerp. Binnen de werkgroep is echter wel de voorkeur uitgesproken voor protocollen uit de TCP/IP suite.

De voorbeelden die in dit document voorkomen hebben betrekking op de toepassing van IVERA-protocol voor de communicatie tussen een verkeersregelinstallatie en een beheerscentrale.

Het IVERA-protocol wordt beheerd door de Stichting Beheer ASTRIN/IVERA-protocol. Voor het gebruik van het IVERA-protocol tussen een verkeersregelinstallatie en een beheerscentrale worden door de stichting licenties uitgegeven. Door de stichting zal een testsuite worden ontwikkeld voor de certificering van de implementaties in zowel beheerscentrales als verkeersregelinstallaties.

2.1 Afkortingen

ASTRIN	Association of Traffic Industries in the Netherlands
BNF	Backus-Naur Form
CVN	Contactgroep Verkeersregeltechnici Nederland
IVER	Initiatiefgroep Verkeersregeltechnici Rijkswaterstaat en Provincies
IVERA	IVER/ASTRIN
OSI	Open Systems Interface
pincode	Persoonlijke Identificatie Nummer-code
PSTN	Public Switched Telephone Network
TCP/IP	Transport Communicatie Protocol / Internet Protocol
UIC	User Identification Control
UNIX	
VRI	VerkeersRegelInstallatie

2.2 Terminologie

	Verwijzing
Object	par. 3.5. Object Definitie
Index-object	par. 3.6. Object soorten
Base-object	par. 3.6. Object soorten
Commando-object	par. 3.6. Object soorten
Event-object	par. 3.6. Object soorten
Gebruiker	par. 3.7. Gebruikers
Gebruikers-groep	par. 3.7. Gebruikers
Master	par. 3.2. Master-Slave
Slave	par. 3.2. Master-Slave
Toepassing	par. 3.4. Toepassingen, Applicaties en Automaten
Applicatie	par. 3.4. Toepassingen, Applicaties en Automaten
Automaat	par. 3.4. Toepassingen, Applicaties en Automaten

2.3 Aanpassing van dit document

De functionele specificatie voor versie 3.01 komt in grote lijnen overeen met de specificatie voor versie 2.10 met het later verschenen addendum.

Het addendum van versie 2.10 is verwerkt in de documentatie voor versie 3.01. Wijzigingen m.b.t het addendum zijn geel gemarkeerd.

Nieuwe functionaliteiten en aanvullingen zijn blauw gemarkeerd.

3. IVERA-protocol

3.1 Inleiding

Het IVERA-protocol is een master-slave protocol op laag 7 (applicatie) van het OSI-model. Via het IVERA-protocol kan een master (beheerscentrale) objecten in een slave (VRI) lezen en schrijven. Een object is gedefinieerd als:

- *Een object binnen het IVERA-protocol is iets dat je kunt selecteren en manipuleren als een eenheid.*
- *Om een object te kunnen selecteren heeft ieder object een unieke naam.*
- *Alle data van een object is van hetzelfde type.*

De volgende tabel geeft een aantal voorbeelden van objecten:

Object	Omschrijving
TGL	Een object dat de geeltijden van alle signaalgroepen bevat.
TOR	Een object dat alle ontruimingstijden bevat.
SG.I	Een object dat de functionele namen van alle signaalgroepen bevat.
VRI.LA	Een object dat alle programma-events bevat die nog niet door de master zijn bevestigd.
VRI.C	Een object waarmee de alarmen van een slave gereset kunnen worden.

Tabel 3.1. Voorbeelden van objecten

De beschrijving van het IVERA-protocol is als volgt opgebouwd:

Paragraaf	Inhoud
Master-Slave	Beschrijving van het gebruikte master-slave principe.
OSI-model	Beschrijving van de plaats van het IVERA-protocol in het 7 lagen OSI-model en de eisen die gesteld worden aan onderliggende netwerklagen.
Toepassing, Applicaties en Automaten	Beschrijving van deze terminologie.
Object definitie	Een algemene definitie van het begrip object.
Object kenmerken	Een aanvulling op de objectdefinitie in de vorm van een aantal objecten met bijzondere kenmerken.
Gebruikers	Een definitie van de begrippen gebruiker en gebruikersgroep
Bericht definitie	Een beschrijving van het protocol tussen de IVERA master en slave

Tabel 3.2. Opbouw beschrijving IVERA-protocol

3.2 Master-Slave

Het IVERA-protocol kent een master (de beheerscentrale) en een slave (de verkeersregelinstallatie).

Opbouwen van een verbinding
Zowel de master als de slave kunnen het initiatief nemen tot het opbouwen van een verbinding. Nadat de verbinding tot stand is gekomen is er sprake van een master-slave protocol.

Opvragen van informatie
Een master kan informatie opvragen uit de slave door het lezen van objecten.

Schrijven van informatie
Een master kan o.a. een parameterinstelling in de slave wijzigen door het schrijven van de nieuwe waarde naar het bijbehorende object in de slave.

Geven van commando's
Een master kan een commando sturen naar een slave door het schrijven naar een object. De betekenis van het commando is in de definitie van het object opgeslagen.

Een gebeurtenis (event) in de slave
Bij een event in de slave waarover de master geïnformeerd moet worden zal de slave eerst een verbinding opbouwen met de master. Als de verbinding aanwezig is zal de slave een event bericht <BerichtSlaveTrigger> versturen naar de master. Een eventbericht dat autonoom van de slave naar de master wordt verstuurd bevat niet de eventdata, maar is slechts een trigger dat een gebeurtenis is opgetreden. Na het ontvangen van een eventbericht is het de taak van de master om de juiste informatie uit de slave te lezen.

3.3 OSI-model

Het IVERA-protocol bevindt zich op laag 7 “de applicatie laag” in het OSI model. Het IVERA-protocol communiceert met onderliggende netwerklagen via standaard functies (stream of file I/O). Het IVERA-protocol doet de volgende aannames, ten aanzien van onderliggende interfacelagen.

- Onderliggende interfacelagen zorgen voor het opzetten en in stand houden van een “connection oriented” verbinding tussen IVERA-master en IVERA-slave. Een mogelijke invulling van zo’n verbinding is een verbinding op basis van TCP/IP sockets.
- Onderliggende interfacelagen dragen er zorg voor dat de bytes verstuurd door de IVERA-master foutvrij en in dezelfde volgorde aankomen bij de IVERA-slave en visa versa.
- Onderliggende interface lagen dragen zorg voor segmentering en routering.
- Indien datacompressie gewenst dan wel noodzakelijk is, dient dit in lagere interface lagen geïmplementeerd te worden.
- Indien data-encryptie noodzakelijk is, dient dit in lagere interface lagen geïmplementeerd te worden.
- Het IVERA-protocol werkt ten minste over TCP/IP en PPP en eventueel ook rechtstreeks op een fysieke verbinding.

3.4 Toepassingen, Applicaties en Automaten

Het IVERA-protocol is geschikt voor de meerdere toepassingen, zoals o.a.

- Communicatie tussen een beheerscentrale en een verkeersregelinstallatie,
- Communicatie met toeritdoseertoestellen,
- Aansturing van borden in een parkeerverwijzingssysteem,
- Koppeling tussen beheerscentrales onderling.

Voor een toepassing is er een unieke set van objecten. De naamgeving van de objecten binnen een toepassing is afgeleid van de voor die toepassing gebruikte naamconventie.

Een object is uniek gedefinieerd met: <TID, ObjectNaam>
waarbij: TID = toepassing identificatie

Binnen een toepassing kunnen er verschillende applicaties bestaan zoals bij een verkeersregelinstallatie o.a. C-COL en RWS-C. Binnen een applicatie staat het vrij zelf objecten te definiëren. De naam van een applicatiespecifiek object begint altijd met de letter ‘Y’.

Een applicatiespecifiek object is uniek gedefinieerd met: <TID, YID, ObjectNaam>
waarbij: YID = applicatie-identificatie.

Binnen een toepassing kunnen er verschillende hardwareplatforms (automaten) bestaan, ieder met hun eigen specifieke mogelijkheden. De naam van een automaat specifiek object begint altijd met de letter ‘X’.

Een automaatspecifiek object is uniek gedefinieerd met: <TID, XID, ObjectNaam>
waarbij: XID = automaatidentificatie.

De achtergrond van deze opzet is dat ontwikkelaars van toepassingen, applicaties en automaten, zelfstandig hun objecten kunnen definiëren, wat het beheer van objecten sterk vereenvoudigt.

NB. Uiteraard is het streven om per toepassing de objecten zoveel mogelijk te standaardiseren.

3.5 Objectdefinitie

Een object zoals gedefinieerd in een IVERA-slave bestaat uit:

- unieke objectnaam
- objectattributen
- data-elementen

De objectnaam bevat een unieke naam voor het object. De objectattributen bevatten alle kenmerken van het object. De data-elementen bevatten de in het object opgeslagen data. De data-elementen in een object zijn opgeslagen als een één of meer dimensionale array. Per object is het aantal dimensies en het aantal data-elementen per dimensie instelbaar. Om praktische redenen is het aantal dimensies per object begrensd op 3.

```
Object
= Naam
+ Omschrijving
+ Type
+ UIC { UIC }
+ Logboek
+ Wijzigingsteller
+ Aantal data-elementen { Aantal data-elementen }
+ Index data-elementnaam { Index data-elementnaam }
+ minimum waarde data-element
+ maximum waarde data-element
+ Index data minimum elementwaarde
+ Index data maximum elementwaarde
+ Index data-element type
+ Data-element formaat
+ Data-element stapgrootte
+ Data-element waarde { Data element waarde }
+ Overzicht alle attributen
```

BNF 1. Object definitie

Alle attributen van een object hebben een unieke naam. Deze attribuutnamen zijn weergegeven in de volgende tabel. Het type geeft aan het datatype van het attribuut; getal of tekst.

Attribuut	Type	Omschrijving
N	1	Naam
O	1	Omschrijving
T	0	Type
U	0	User Identificatie Control
L	0	Logboek
W	0	Wijzigingsteller
E	0	aantal data-elementen
I	1	Index verwijzing per dimensie
MIN	0	Minimum data-elementwaarde
MAX	0	Maximum data-elementwaarde
IMIN	1	Index data-element minimumwaarde
IMAX	1	Index data-element maximumwaarde
ITYPE	1	Index data-element type
F	0	Data-element formaat
S	0	Data-element stapgrootte
A	1	Overzicht alle attributen

Tabel 3.3. Object attributen

In het geval van meerdere dimensies worden het aantal data elementen en de index verwijzingen weergegeven door middel van een volgnummer:

- E1=aantal data elementen dimensie 1, E2=aantal data elementen dimensie 2, etc.
- I1= index dimensie 1, I2=index dimensie 2, etc.

```
N       = ObjectNaam
O       = String
T       = 0 | 1 | 2
U       = Groep4 * 1000 + Groep3 * 100 + Groep2 * 10 + Groep1
L       = 0 | 1
W       = PosIntegerWaarde
E       = PosIntegerWaarde
I       = ObjectNaam
MIN     = IntegerWaarde
MAX     = IntegerWaarde
IMIN    = ObjectNaam
IMAX    = ObjectNaam
ITYPE   = ObjectNaam
F       = PosIntegerWaarde
S       = PosIntegerWaarde
A       = /* Overzicht alle attributen */

Groep1 = Groep
Groep2 = Groep
Groep3 = Groep
Groep4 = Groep
Groep  = 0 | 4 | 6

IntegerWaarde = ["-"] PosIntegerWaarde
PosIntegerWaarde = DecWaarde
DecWaarde = Digit { Digit }
Digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

ObjectNaam = Letter { Eletter } ["."] { Eletter }
Letter = "A" .. "Z" | "a" .. "z"
Eletter = Letter | Digit
```

BNF 2. Attributen definitie

NB. Integer waarden zijn getallen in het bereik -2^{31} tot $+2^{31}-1$.

Naam
Ieder object heeft een unieke naam. Een naam bestaat uit een combinatie van letters, cijfers en eventueel één punt. Een naam begint altijd met een letter. Een naam kent geen onderscheid tussen hoofdletters en kleine letters. Een naam bestaat uit maximaal 16 karakters.

Referentie	Omschrijving
TGL	Geeltijden
TOR	Ontruimingstijden
P	Parameters
LSGE	Externe signaalgroep status (tekststring)
SG.I	Interne signaalgroep status (tekststring)

Tabel 3.4 Voorbeelden van objectnamen.

Algemene richtlijnen voor de naamgeving van objecten.
- Een naam bestaat uit een unieke combinatie van letters en cijfers.
- Een naam begint met een letter.
- Een punt (.) is bedoeld voor groepsindelingen of categorieën.

Door de indeling van objecten in groepen kan het zoeken van objecten die een relatie met elkaar hebben worden geautomatiseerd. Bijv. “SG.*” levert alle aan signaalgroepen gerelateerde objecten.

NB. Voor meer informatie over objectnaamgeving zie paragraaf “Beheer van objecten”.

Omschrijving
Een omschrijving van het object in leesbare tekst. De omschrijving bevat maximaal 32 karakters.

Type
Het type beschrijft het formaat van de data-elementen in het object.

Type	Omschrijving
0	Getal (32 bits integer)
1	Tekststring
2	Cluster (optie)

Tabel 3.5. Type objecten

Een object met het type 0 (getal) wil zeggen, dat de data-elementen van het object bestaan uit getallen. In het geval dat de data-elementen tekststrings bevatten is het object van het type 1 (tekststring).

Aantal data-elementen
Het actuele aantal data-elementen in een object. In het geval van een object met meerdere dimensies (bijv. een matrix) wordt het aantal data-elementen per dimensie gespecificeerd; het totale aantal data elementen van het object is dan een vermenigvuldiging van het aantal data-elementen van iedere dimensie.

Een object kan 0 tot maximaal 2^{16}(65536) data-elementen bevatten. In het geval dat een object 0 elementen bevat is het object ‘leeg’. Indien een ‘leeg’ object wordt benaderd, zal de slave antwoorden met een foutcode ERR_EMPTY (zie tabel 3.11).
De nummering van de data-elementen begint bij 0 en loopt tot 2^{16}-1, d.w.z. met 0 wordt het 1^{st} data-element aangeduid en met 2^{16}-1 het laatste data-element.

Normaliter is het aantal elementen in een object constant; dit is echter geen eis. Een voorbeeld van een object waarvan het aantal elementen varieert, is een eventbuffer. In een eventbuffer correspondeert het aantal data-elementen met het aantal events in het buffer. Het aantal elementen kan in dit geval ook 0 zijn.

Tevens is het mogelijk en in sommige gevallen zelfs wenselijk om een object met 0 elementen te definiëren. Een voorbeeld van een object dat 0 elementen kan bevatten is het object P (parameters). In het geval dat een IVERA-slave geen parameters heeft, zijn er twee keuzen; het object P niet definiëren of een object P met 0 elementen definiëren. In dit geval geniet het de voorkeur een object P met 0 elementen te definiëren, om zo te accentueren dat de IVERA-slave het object P wel ondersteunt, maar dat er geen parameters zijn.

UIC (User Identification Control)
Het UIC attribuut is een masker dat per gebruikersgroep aangeeft welke toegangsrechten een groep van gebruikers heeft op de data-elementen van een object. Hiervoor is per gebruikersgroep een bitmasker gedefinieerd dat aangeeft wat de lees-, schrijf- en executierechten van de gebruikersgroep zijn. De indeling van het UIC bitmasker komt overeen met de gebruikersrechten binnen UNIX.

Waarde	bitmask	Omschrijving
0	000	geen rechten
1	001	alleen executeren
2	010	alleen schrijven
3	011	executeren en schrijven
4	100	alleen lezen
5	101	executeren en lezen
6	110	lezen en schrijven
7	111	executeren, lezen en schrijven

Tabel 3.6. Gebruikersrechten

Daar het IVERA-protocol alleen lezen en schrijven ondersteunt, zijn alleen de volgende opties van belang:

UIC	Omschrijving
0	geen rechten
4	alleen lezen
6	lezen en schrijven

Tabel 3.7. Gebruikersrechten binnen IVERA

Voorbeeld: UIC = 6664
- Groep1 = 4 (alleen lezen)
- Groep2, groep3 en groep4 = 6 (lezen en schrijven)

NB. Hierbij is de aanname gedaan dat iemand die mag schrijven ook mag lezen.
NB. Zie paragraaf "Gebruikers" voor een beschrijving van de gebruikersgroepen.

Index data-elementnamen
Een index bevat een Naam (zie BNF ObjectNaam) die verwijst naar een ander object. Een index biedt de mogelijkheid om de elementen van een object van een logische naam te voorzien. De logische namen zijn opgeslagen in de data-elementen van het object waar de index heen verwijst. In een object kan per dimensie een index worden opgegeven.

Een voorbeeld is de index van het object TGL (geeltijden) die verwijst naar het object SG.I (signaalgroepnamen).

Logboek
Voor ieder object is vastgelegd of wijzigingen van de data-elementen moeten worden opgeslagen in het parameterlogboek.

Wijzigingsteller (optioneel)

De wijzigingsteller van een object is bedoeld als een variabele die wordt verhoogd, als een van de data-elementen in een object wijzigt. Deze vlag zou handig kunnen zijn voor objecten met een groot aantal data-elementen die slechts sporadisch van toestand veranderen. De wijzigingsteller voorkomt dat een master die geïnteresseerd is in de wijzigingen, regelmatig alle data elementen moet lezen.
Aangezien de wijzigingsteller in de praktijk niet wordt gebruikt, is het niet voorgeschreven deze met een zinnige waarde te vullen. Wel blijft het attribuut om compatibiliteitsredenen gehandhaafd.

Minimumwaarde en maximumwaarde van data-elementen
Voor ieder object is een algemene minimum- en maximumwaarde instelbaar die geldt voor alle data-elementen. Iedere waarde die naar een data-element van het object wordt geschreven moet aan de volgende voorwaarde voldoen:

minimumwaarde <= waarde <= maximumwaarde

NB. De attributen minimum- en maximumwaarde zijn alleen van toepassing op objecten van het type 0 (getal).

NB. Voor een object van het type 1 (tekststring) corresponderen de minimum- en maximumwaarde met de minimale en maximale lengte van de string.

Indexminimum en -maximum van data-elementen
Voor iedere getalobject is het mogelijk een ander object te gebruiken als minimum of maximum. Dit biedt de mogelijkheid om per data-element een minimum- en maximumwaarde te specificeren. De verwijzing is een naam (zie BNF ObjectNaam) die verwijst naar een object.

Een voorbeeld is het TGGL (garantiegeeltijd) object dat als indexminimum aan het TGL (geeltijd) object is gekoppeld. Op deze manier wordt voorkomen dat een geeltijd onder de garantiegeeltijd ingesteld wordt.

NB. De attributen index minimum- en maximumwaarde zijn alleen van toepassing op objecten van het type 0 (getal).

Indextype van data-elementen
Voor ieder getalobject is het mogelijk een ander object te gebruiken als type. Dit biedt de mogelijkheid om per data-element een type te specificeren. De verwijzing is een naam (zie BNF ObjectNaam) die verwijst naar een object.

Formaat data-element
Per object is er de mogelijkheid één dataformaat te definiëren. Het dataformaat wordt aangeduid door middel van een getal. De betekenis van het dataformaat is afhankelijk van de toepassing.

Stapgrootte data-element
Per object is een stapgrootte gedefinieerd. De stapgrootte geeft aan met welke stapgrootte een waarde in een data-element van het object kan worden ingesteld. Bij het schrijven van data naar een object worden alleen die waardes geaccepteerd die een veelvoud zijn van de stapgrootte; alle tussenliggende waardes worden geweigerd.

NB. Het attribuut stapgrootte is alleen van toepassing op objecten van het type 0 (getal).

NB. Afwijkende reeksen zoals 1, 3, 5,7 kunnen niet als attributen worden gespecificeerd, wel is het mogelijk deze beperkingen aan te brengen door deze hard in de code van het object te programmeren.

Waarde data-element
Een object kan 0 tot 2^{16} data elementen bevatten.

Overzicht alle attributen

Per object is het mogelijk om via het attribuut ‘A’ alle attributen in 1 keer te lezen of te wijzigen. Het attribuut ‘A’ is een tekst string met het volgende formaat:

```
Overzicht alle attributen = AttribuutDef { “,” AttribuutDef }
AttribuutDef = AttribuutNaam “=“ AttribuutWaarde
AttribuutNaam = “N” | “O” | “U” | “L” | “W” | “E” | “E1” | “E2” | “I” | “I1” | “I2” | “MIN” | “MAX” | “IMIN” | “IMAX”
| “F” | “S” | “A” | “T”

AttribuutWaarde = IntegerWaarde | AttribuutString1 | AttribuutString2
AttribuutString1= ““ { Karakter1 } ““
AttribuutString2= Karakter2 { Karakter2 }
Karakter1 = “A”..”Z” | “a”..”z” | “0”..”9” | “ “ | “,” | “.”
Karakter2 = “A”..”Z” | “a”..”z” | “0”..”9” | “.”
```

BNF 3. Overzicht alle attributen

Als alle attributen van een object worden opgevraagd dan dienen minimaal de volgende attributen te worden vermeld:

N	Objectnaam
T	Objecttype
F	Formaat
E	Aantal elementen (of indien meer dimensionaal dan E1 en E2).
U	Toegangsrechten

NB. De waarde van de attributen A en DATA worden uiteraard nooit vermeld.
NB. Voor voorbeelden zie tabel 3.8.

3.6 Objectsoorten

In deze paragraaf worden globaal een aantal soorten objecten en hun kenmerken beschreven. Deze beschrijving is een aanvulling op de formele objectdefinitie (zie paragraaf “Objectdefinitie”).

Base-object

Een *base-object* is een object van het type 1. Een *base-object* bevat een lijst met namen van de in de IVERA-slave aanwezige objecten van een bepaald type. Het aantal data-elementen van een *base-object* komt overeen met het aantal objecten van een bepaald type.

De *base-objecten* “BB0” t/m “BB99” leveren de namen van de objecten van een bepaald type. Het volgnummer komt overeen met het objecttype.

De *base-objecten* “BBA0” t/m “BBA99” leveren alle attributen van de objecten van een bepaald type.

In een IVERA-slave met 4 objecten van het type getal (TGL, TGGL, TOR, TGOR) bevatten de objecten BB0 en BBA0 beide 4 data-elementen.

Master	Slave antwoord
BBA0	BBA0=“N=TGL,T=0, E=4,L=1,U=6664,I=SG.I,S=1,MIN=2,MAX=10,IMIN=TGGL,O=‘Geeltijd’”, “N=TGGL,T=0,E=4,L=0,U=4444,I=SG.I,S=1,MIN=0,MAX=10,O=‘Garantiegeeltijd’”, “N=TOR,T=0,E1=4,E2=4,L=1,U=6664,I1=SG.I,I2=SG.I,S=1,MIN=-1,MAX=10,IMIN=TGOR, O=‘Ontruimingstijd’”, “N=TGOR,T=0,E1=4,E2=4,L=0,U=4444,I1=SG.I,I2=SG.I,S=1,MIN=-1,MAX=10, O=‘Garantieontruimingstijd’”
BB0	BB0=”TGL”,”TGGL”,”TOR”,”TGOR”

Tabel 3.8. Voorbeeld van objecten BB0 en BBA0.

Index object

Een *indexobject* is een object van het type 1. In een *indexobject* bevatten de data-elementen logische namen voor de data-elementen in een ander object.

Een voorbeeld is het object “SG.I” (signaalgroepnamen). Dit object kan als index dienen voor andere signaalgroep georiënteerde objecten, zoals het object TGL” (geeltijden).

Commando-object
Vanuit een IVERA-master kunnen via *commando-objecten* commando’s naar een slave worden gestuurd. Het sturen van een commando is geïmplementeerd als het schrijven van een waarde naar een data-element van een object. De combinatie van object, data-element en waarde bepaalt het commando dat wordt gegeven.

Een voorbeeld is het object “VRI.C”. Door het schrijven van een waarde naar dit object kunnen specifieke alarmen/events in een slave worden gereset.

Event-object
Een event is een gebeurtenis in de slave. Deze events worden door de slave opgeslagen in een zogenaamd *event-object*. De master kan de opgetreden events ophalen door het lezen van het bijbehorende object. Nadat de master een event heeft gelezen, kan de master het gelezen event bevestigen. Een *event-object* wordt gekenmerkt door het feit dat het aantal data-elementen niet constant is. Het aantal data-elementen komt overeen met het aantal events in het *event-object*.

In de slave wordt er onderscheid gemaakt tussen twee *event-objecten*;
- Logboek (.LB); dit object bevat de laatste *n* events. Waarbij element 0 overeenkomt met het nieuwste event en element *n-1* met het oudste event. Als het eventbuffer vol is, vervalt het oudste element bij het toevoegen een nieuw event.
- Onbevestigde events (.LA); dit object bevat de nieuwste *m* onbevestigde events. Waarbij element 0 overeenkomt met het oudste onbevestigde event en element *m-1* met het nieuwste onbevestigde event. De master kan een event bevestigen door het schrijven van een willekeurige waarde naar een data-element. Na bevestiging wordt het event door de slave uit het object verwijderd.

.LA

Event6
Event7
Event8
Event9
Nieuw

Eventbuffer

Nieuw
Onbevestigde events
Bevestigde events
Oud

.LB

0	Nieuw
0	Event9
0	Event8
0	Event7
0	Event6
1	Event5
1	Event4
1	Event3
1	Event2
1	Oud

Figuur 3.1. Event-objectdefinitie

De bedoeling van een event-object (LA) is dat een event ten minste een keer aan de master wordt gemeld. Nadat de master het event heeft gelezen zal de master het event bevestigen. Indien er tussen het lezen en het bevestigen door de master een nieuw event ontstaat, dan gaat dit niet verloren, doordat alleen de gelezen events worden bevestigd.
Tevens kan een event alleen worden bevestigd, als alle voorgaande events zijn bevestigd.

Een voorbeeld van het lezen en bevestigen van events: In de slave zijn 5 events aanwezig in het event object “VRI.LA”, te weten “melding 1” t/m “melding 5”. De master kan alle aanwezige

events lezen door het lezen van alle elementen van het object “VRI.LA”. De slave antwoord hierop met alle beschikbare events.

Master	Slave
VRI.LA	VRI.LA=”melding 1”,”melding 2”,”melding 3”,”melding 4”,”melding 5”

Tabel 3.9.1. Voorbeeld lezen van event object.

Nadat de master heeft gelezen, maar voor dat de master de events heeft bevestigd, ontstaat er een nieuw event; “melding 6”. Het object VRI.LA bevat nu dus 6 elementen.

Onbevestigde events
”melding 1”,”melding 2”,”melding 3”,”melding 4”,”melding 5”,”melding 6”

Tabel 3.9.2. Data in object VRI.LA

De master kan de 5 gelezen events bevestigen door het schrijven van willekeurige data naar de gelezen elementen.

Master	Slave
VRI.LA/#0-#4=””	:A

Tabel 3.9.3. Voorbeeld bevestigen van events in een event object.

Na het bevestigen van de 5 gelezen events bevat het object nog 1 element.

Onbevestigde events
”melding 6”

Tabel 3.9.4. Data in object VRI.LA

Het nieuwe event, “melding 6” is niet verloren gegaan, doordat alleen de gelezen events worden bevestigd. Het commando bevestigen van alle events (VRI.LA/*=””) wordt niet gebruikt, omdat dit tot gevolg heeft, dat ook alle nog niet gelezen events bevestigd worden.

Om de master in geval van veel events te waarschuwen is er een eventreeks 2510, 2511, 2512 en 2513 om te waarschuwen dat een logboek vol dreigt te raken.

NB. Indien een event-object vol is kunnen er events verloren gaan.
NB. Het is de taak van de master om regelmatig de informatie in het object te lezen.

3.7 Gebruikers

Een gebruiker is iets of iemand die via het IVERA-protocol toegang heeft tot de objecten van de IVERA slave. Om het systeem eenvoudig te houden kent een IVERA-slave geen individuele gebruikers, maar alleen gebruikersgroepen. Een gebruikersgroep is een verzameling van gebruikers met dezelfde rechten.

Een IVERA-slave ondersteunt 4 gebruikersgroepen, waarvan de rechten per object instelbaar zijn.

De volgende gebruikersgroepen zijn gedefinieerd:

Groep	pincode	Omschrijving
1	xxxx	Gebruikersgroep 1
2	xxxx	Gebruikersgroep 2
3	xxxx	Gebruikersgroep 3
4	xxxx	Gebruikersgroep 4

Tabel 3.10. Gebruikersgroepen

Inloggen
De gebruikersgroepen hebben een eigen pincode waarmee een gebruiker kan inloggen. Het inloggen gebeurt door het schrijven naar het object “LOGIN”. De pincodes worden eenmalig bepaald en zijn niet instelbaar. De pincodes voor het inloggen zijn klantspecifiek.

Indien de ingevoerde pincode onjuist is wordt door de IVERA-slave de foutcode :E=16 geretourneerd. Na 3x foutief inloggen wordt de communicatieverbinding volledig - dat betekent bij kiesverbindingen met inbegrip van de fysieke verbinding - door de VRI verbroken. Voordat de verbinding wordt verbroken is het mogelijk, maar niet verplicht, dat foutcode :E=16 wordt verstuurd.

Indien een inlogpoging wordt gedaan voor een ander niveau dan waarop reeds is ingelogd en deze poging faalt, dan blijft de gebruiker op het huidige inlogniveau. Mocht dit drie keer achter elkaar mislukken dan zal de verbinding worden verbroken.

Alleen de objecten LOGIN en PING kunnen zonder inloggen worden gelezen dan wel geschreven.

uitloggen
Als er is ingelogd als gebruikersgroep, kan erop de volgende manieren worden uitgelogd.
- Bij het verbreken van de verbinding tussen master en slave.
- Bij een time-out van ½ uur, waarin er geen activiteit is vanuit de master.
- Door het commando “LOGIN/#0=0”.

NB: Bij het uitlezen van het object “LOGIN” is het resultaat ongedefinieerd.

3.8 Berichtdefinitie

Voor het lezen of schrijven van een object stuurt een master een bericht in leesbare tekst naar de slave. Het bericht wordt afgesloten met een return (karakter code 13 decimaal). De slave antwoordt met een bericht in leesbare tekst, afgesloten met een return.

```
Bericht = [BerichtID] (BerichtMaster | BerichtSlave) CarriageReturn
BerichtID = “@” PosIntegerWaarde “#”
BerichtMaster = ObjectRef [ “=“ ArgumentLijst ]
BerichtSlave = BerichtSlaveErr | BerichtSlaveAck | BerichtSlaveAntw |
          BerichtSlaveAckHand | BerichtSlaveAntwHand | BerichtSlaveTrigger

BerichtSlaveErr = “:E=“ SlaveErrCode
BerichtSlaveAck = “:A”
BerichtSlaveAntw = “=“ ArgumentLijst
BerichtSlaveAckHand = ObjectRef “=“ ArgumentLijst
BerichtSlaveAntwHand = ObjectRef “=“ ArgumentLijst
BerichtSlaveTrigger = “:T=” TriggerCode

TriggerCode = PosIntegerWaarde
SlaveErrCode = “0” | “1” | “10” | “11” | “12” | “13” | “14” | “15” | “16” | “17” | “18” | “19”
CarriageReturn = Karakter code 13 decimaal

ObjectRef = ObjectNaam [“:” AttribuutNaam ] | ([ “/” ElementBereik { “,” ElementBereik } ] )
ElementBereik = “*” | Bereik
Bereik = Element [ “-“ [ Element ] ]
Element = (“#” PosIntegerWaarde) | IndexNaam

IndexNaam = Eletter { Eletter | “_” }
ArgumentLijst = Argument { “,” Argument }
Argument = IntegerWaarde | TekstString
TekstString = DubbelQuote { DetailString } DubbelQuote
```

DubbelQuote = “““
Komma = “,”
AsciiKarakter = Karakter uit de ASCII karakterset met waarde tussen 31 en 127 met uitzondering van DubbelQuote (ASCII 34) en Komma (ASCII 44)
AsciiString = { AsciiKarakter }
DetailString = { AsciiString | Komma }
leeg = “”

BNF 4. Berichtdefinitie.

NB. Het IVERA-protocol maakt geen onderscheid tussen hoofdletters en kleine letters. In zowel objectnamen als attribuutnamen mogen hoofdletters en kleine letters door elkaar gebruikt worden.

NB. Als het bericht vanuit de master een <BerichtId> bevat dient in het antwoord van de slave hetzelfde <BerichtId> te worden teruggestuurd.

NB. Het ondersteunen van een combinatie van indexnamen en indexnummers in één bericht is optioneel (zie definitie Element).

SlaveErrCode	Code	Reden	
ERR_ILLEGAL	0	Bericht voldoet niet aan het IVERA bericht formaat.	R/W
ERR_OVERFLOW	1	De slave heeft onvoldoende geheugen om het bericht te kunnen verwerken.	R/W
ERR_OBJECT	10	Object is niet gedefinieerd in de slave.	R/W
ERR_USER	11	Master heeft geen autorisatie om het object te lezen/schrijven.	R/W
ERR_RANGE	12	Het gespecificeerde element bereik is ongeldig.	R/W
ERR_INDEX	13	De slave kan een of meer opgegeven indices niet vertalen.	R/W
ERR_DIM	14	Indien niet alle dimensies zijn gespecificeerd.	W
ERR_WRANGE	15	Het aantal elementen komt niet overeen met het aantal data argumenten.	W
ERR_DATA	16	De data in de argumentlijst is ongeldig.	W
ERR_EMPTY	17	Het aantal elementen van het object is 0.	R/W
ERR_STEP	18	Waarde is geen veelvoud van de stapgrootte.	W
ERR_ATTRIB	19	Het opgegeven attribuut is ongeldig.	R/W

Tabel 3.11. IVERA-slave foutcodes.

3.8.1 Object Elementbereik

Een object bestaat uit 1 of meer data elementen. Een object is gedefinieerd als een meerdimensionale array van data-elementen, waarbij het aantal dimensies in theorie onbeperkt is. Voorbeelden van objecten met 2 dimensies zijn o.a. de ontruimingstijdenmatrix en plan- (c.q. programma-)afhankelijke maximumgroentijden.

In het IVERA-protocol is een bereik van data-elementen gedefinieerd als:

bereik van data-elementen = [“/” ElementBereik { “,” ElementBereik }]
ElementBereik = “*” | Bereik
Bereik = Element [“-“ [Element]]
Element = (“#” PosIntegerWaarde) | Naam

Bij een object met meer dan 1 dimensie is het bereik gedefinieerd als:

ElementBereik $_{\text{dimensie 1}}$, ElementBereik $_{\text{dimensie 2}}$, .. , ElementBereik $_{\text{dimensie n}}$

Vertaald in x en y coördinaten komt dit overeen met:

y_1-y_2,x_1-x_2

Het bereik dient te voldoen aan de volgende algemene voorwaarden:

- Geen bereikspecificatie komt overeen met **alle** elementen.
- De nummering van de data-elementen begint bij 0.

- Een bereikdefinitie is ongeldig als:
    - Een van de elementen buiten bereik is.
    - Indien het eerste elementnummer groter is dan het tweede elementnummer.
    - Een naam niet (via een *index-object*) vertaald kan worden in elementnummer.
- Voor schrijfoperaties gelden de volgende aanvullende voorwaarden:
    - Voor iedere dimensie moet het bereik volledig gespecificeerd zijn.
    - Indien de argumentlijst 1 argument bevat, wordt dit argument naar alle elementen in het gespecificeerde bereik geschreven.
    - Indien de argumentlijst meerdere argumenten bevat, moet het aantal argumenten overeenkomen met het aantal elementen in het gespecificeerde bereik.

In de volgende paragrafen zal het specificeren van een element bereik worden toegelicht aan de hand van twee voorbeelden:
1. Geeltijden van een signaalgroep (TGL)
2. Ontruimingstijdenmatrix (TOR)

De VRI in de voorbeelden heeft 4 signaalgroepen. De namen van deze signaalgroepen zijn opgeslagen in het *index object* “SG.I”.

Element	Index
0	SG01
1	SG02
2	SG03
3	SG04

Tabel 3.12. Index object “SG.I”

3.8.1.1 Array (1 dimensionaal)

De mogelijkheden voor het adresseren van de geeltijden van de signaalgroepen zijn weergegeven in de volgende tabel.

Methode	Omschrijving
TGL	Alle geeltijden
TGL/*	Alle geeltijden (gelijk aan TGL)
TGL/#0	Geeltijd van element 0 (SG01)
TGL/#0-#3	elementen 0, 1, 2, 3
TGL/#2-	alle elementen vanaf 2 (2, 3)
TGL/SG01	Geeltijd SG01
TGL/SG01-SG04	Geeltijden van SG01 t/m SG04
TGL/SG03-	Geeltijden van SG03 en SG04
TGL/#1-SG04	Geeltijden SG02, SG03, SG04

Tabel 3.13. Voorbeeld lezen van objecten

3.8.1.2 Matrix (2 dimensionaal)

Het specificeren van het elementbereik voor een matrix wordt verklaard aan de hand van de ontruimingstijdenmatrix van een VRI. De volgende tabel bevat de ontruimingstijdenmatrix voor een VRI met 4 signaalgroepen. De waarden in de tabel corresponderen met denkbeeldige elementnummers.

	SG01	SG02	SG03	SG04
SG01	0	1	2	3
SG02	4	5	6	7
SG03	8	9	10	11
SG04	12	13	14	15

Tabel 3.14. Voorbeeld object met 2 dimensies

De mogelijkheden voor het adresseren van de ontruimingstijden zijn weergegeven in de volgende tabel.

Methode	Omschrijving
TOR	Alle ontruimingstijden
TOR/*	Alle ontruimingstijden (gelijk aan TOR)
TOR/*,*	Alle ontruimingstijden (gelijk aan TOR)
TOR/SG03,SG02	element 9
TOR/SG01,*	elementen 0, 1, 2, 3
TOR/SG01	gelijk aan TOR/SG01, *
TOR/*,SG02	elementen 1, 5, 9, 13
TOR/SG01-SG03,SG01	elementen 0, 4, 8
TOR/SG02,SG02-SG03	elementen 5, 6
TOR/SG03,SG02-	elementen 9, 10, 11
TOR/SG01-SG02	elementen 0, 1, 2, 3 en 4, 5, 6, 7

Tabel 3.15. Voorbeelden van het lezen van een object met 2 dimensies

3.9 Data context diagrammen

3.9.1 Lezen van objecten

Vanuit een master kunnen objecten worden gelezen door het specificeren van een objectnaam en bereik van elementen. De slave antwoordt met de gewenste data, indien een geldig object wordt gelezen of met een errorbericht in het geval van een fout.

Master	Slave	Omschrijving
<ObjectRef>	<ObjectRef>=<ArgumentLijst>	Geldig object gelezen.
<ObjectRef>	:E=<SlaveErrCode>	Fout tijdens lezen object
<ObjectRef>		Fout tijdens interpretatie bericht
<BerichtID><ObjectRef>	<BerichtID>=<ArgumentLijst>	Geldig object gelezen.
<BerichtID><ObjectRef>	<BerichtID>:E=<SlaveErrCode>	Fout tijdens lezen object
<BerichtID><ObjectRef>		Fout tijdens interpretatie bericht

Tabel 3.16. Data-contextdiagram voor het lezen van een object

Een slave antwoordt met een errorbericht ":E=", als het bericht vanuit de master niet aan de definitie <ObjectRef> voldoet, of als het bericht wel aan de definitie <ObjectRef> voldoet, maar de slave geen antwoord kan geven op de gestelde vraag.

Master	Slave	Omschrijving
TGL	TGL=3,3,3,3	Geeltijd van alle signaalgroepen
TGL/SG01-SG03	TGL/SG01-SG03=3,3,3	Geeltijd van SG01,SG02,SG03
TOR/SG01	TOR/SG01=-1,2,3,4	Ontruimingstijden van SG01
TOR/SG01,SG02	TOR/SG01,SG02=2	Ontruimingstijd SG01 -> SG02
@1#TGL	@1#=3,3,3,3	Geeltijd van alle signaalgroepen
@2#TGL/SG01-SG03	@2#=3,3,3	Geeltijd van SG01,SG02,SG03
@3#TOR/SG01	@3#=-1,2,3,4	Ontruimingstijden van SG01
@4#TOR/SG01,SG02	@4#=2	Ontruimingstijd SG01 -> SG02

Tabel 3.18. Voorbeelden van het lezen van objecten.

3.9.2 Schrijven van objecten

Vanuit een master kan data naar een object worden geschreven door het specificeren van een objectnaam, een bereik van elementen en een lijst met argumenten. Bij een geldig bericht antwoordt de slave met een acknowlegde (ACK) bericht naar de master. In het geval van een fout antwoordt de slave met een errorbericht.

Master	Slave	Omschrijving
<ObjectRef>=<ArgumentLijst>	<ObjectRef>=<ArgumentLijst>	Geldig object geschreven
<ObjectRef>=<ArgumentLijst>	:E=<SlaveErrCode>	Fout tijdens schrijven object
<ObjectRef>=<ArgumentLijst>		Fout tijdens interpretatie bericht
<BerichtID><ObjectRef>=<ArgumentLijst>	<BerichtID>:A	Geldig object geschreven
<BerichtID><ObjectRef>= <ArgumentLijst>	<BerichtID>:E=<SlaveErrCode>	Fout tijdens schrijven object
<BerichtID><ObjectRef>= <ArgumentLijst>		Fout tijdens interpretatie bericht

Tabel 3.19. Data context diagram voor het schrijven naar een object.

Een slave antwoordt met een errorbericht ":E=" als:

- het bericht niet voldoet aan de definitie <ObjectRef>=<ArgumentLijst>.
- het bereik niet volledig is gespecificeerd.
- het aantal data argumenten niet overeenkomt met het aantal elementen.
- de data ongeldig is.
- de master geen autorisatie heeft om de data te wijzigen.

Master	Slave	Omschrijving
TGL/#0=3	TGL/#0=3	Parameterwijziging geaccepteerd.
TGL/SG02=9	:E=16	NAK_DATA, buiten bereik.
TGL/SG02=4	:E=10	Gebruiker heeft geen schrijf rechten.
TOR/SG01,SG02=2	TOR/SG01,SG02=2	Parameterwijziging geaccepteerd.
PING/#0=5	PING/#0=5	Test verbinding.
@1#TGL/#0=3	@1#:A	Parameterwijziging geaccepteerd.
@2#TGL/SG02=9	@2#:E=16	NAK_DATA, buiten bereik.
@3#TGL/SG02=4	@3#:E=10	Gebruiker heeft geen schrijf rechten.
@4#TOR/SG01,SG02=2	@4#:A	Parameterwijziging geaccepteerd.
@5#PING/#0=5	@5#:A	Test verbinding.

Tabel 3.21. Voorbeelden van het schrijven naar objecten.

3.9.2.1 Schrijven van meerdere elementen

Door middel van een enkel bericht is het mogelijk om meerdere elementen van een object te schrijven. Dit kan op twee manieren:

- 1 waarde naar meerdere data elementen, of
- een aantal waardes naar een aantal opeenvolgende data elementen.

Master	Slave	Omschrijving
@1#TGL=3	@1#:E=14	Bereik is niet volledig gespecificeerd
@2#TGL/*=3	@2#:A	Alle geeltijden naar 3 s.
@3#TGL/SG01-SG02=3	@3#:A	TGL/SG01=3 en TGL/ SG02=3
@4#TGL/SG01-SG03=3	@4#:E=16	Data niet geldig omdat gar. geeltijd SG03=4 s.
@5#TGL/SG01-SG02=3,4	@5#:A	TGL/SG01=3, TGL/SG02=4
@6#TGL/SG01-SG03=3,4	@6#:E=15	Aantal elementen =3, aantal argumenten = 2

Tabel 3.22. Voorbeelden van het schrijven van meerdere data-elementen naar objecten.

Bij de bovenstaande tabel de volgende opmerking:

In het geval dat met 1 commando meerdere elementen worden gewijzigd, kan het gebeuren dat het schrijven naar één van de elementen wordt geweigerd vanwege ongeldige data. De slave handelt in dit geval als volgt:

- De slave antwoordt met ":E=16", indien één van de data-elementen ongeldig is.
- Geen van de data-elementen wordt gewijzigd.
- Dit houdt concreet in, dat de slave voor het schrijven van de data eerst alle data-elementen moet controleren.

NB. Bij een schrijfcommando moet altijd het volledige bereik worden gespecificeerd om te voorkomen dat er per ongeluk verkeerde elementen worden gewijzigd. Dit is met name van belang bij het handmatig ingeven van commando's.

3.9.3 Lezen van objectattributen

Vanuit de IVERA-master kunnen de attributen van een object in de slave worden gelezen. De syntax voor het lezen van de attributen is gelijk aan die voor het lezen van het object data-elementen. Het lezen van attributen kan op de volgende manieren:

- Het lezen van 1 attribuut, of
- Het lezen van alle attributen.

Voor een beschrijving van de attributen wordt verwezen naar de paragraaf “Objectdefinitie”.

Master	Slave	Omschrijving
TGL:N	TGL:N=”TGL”	Objectnaam
TGL:T	TGL:T=0	Objecttype
TGL:E	TGL:E=4	4 signaalgroepen
TGL:U	TGL:U=664	User Identificatie Control
TGL:L	TGL:L=1	Logboek aan
TGL:I	TGL:I=”SG.I”	Index object(s)
TGL:W	TGL:W=1	Wijzigingsteller
TGL:MIN	TGL:MIN=2	Minimumwaarde
TGL:MAX	TGL:MAX=10	Maximumwaarde
TGL:IMIN	TGL:IMIN=”TGGL”	Index minimum
TGL:IMAX	TGL:IMAX=””	Index maximum
TGL:S	TGL:S=1	Stapgrootte
TGL:A	TGL:A=”/* string met attributen */”	Uitlezen alle attributen
TOR:E	TOR:E=4,4	matrix van 4 bij 4.
TOR:I	TOR:I=“SG.I”,”SG.I”	
TOR:IMIN	TOR:IMIN=“TGOR”	

Tabel 3.23. Voorbeelden van het lezen van objectattributen.

NB. Voor de definitie van de string met attributen wordt verwezen naar paragraaf “Objectsoorten”.

3.9.4 Wijziging van objectattributen (optioneel)

Vanuit de IVERA-master kunnen de attributen van een object in de slave worden gewijzigd. De syntax voor het wijzigen van de attributen is gelijk aan die voor het schrijven van het object data-elementen. Het wijzigen van attributen is alleen mogelijk met gebruikersgroep 4 privileges. Het schrijven van attributen kan op de volgende manieren:

- Het schrijven van 1 attribuut, of
- Het schrijven van meerdere attributen.

Master	Slave	Omschrijving
@1#TGL:L=1	@1#:A	Logboek aan
@2#TGL:A=”L=1,IMIN=TGGL”	@2#:A	
SG.I:S=1	:E=19	Stapgrootte wordt niet ondersteund door tekstobjecten.

Tabel 3.24. Voorbeelden van het schrijven van objectattributen.

NB. Indien een attribuut niet wordt ondersteund, antwoordt de IVERA-slave met ERR_ATTRIB.
NB. Voor de definitie van de string met attributen wordt verwezen naar paragraaf “Objectsoorten”.

3.9.5 Master-slave synchronisatie

Bij een handmatige invoer van IVERA-commando’s wordt er geen gebruik gemaakt van berichtnummers.

In het geval van communicatie tussen twee computers, bijvoorbeeld de communicatie tussen een beheerscentrale en een verkeersregelinstallatie, wordt gebruik gemaakt van berichtnummers. De master genereert de berichtnummers, die door slave transparant worden terugzonden. De master vergelijkt het antwoord van de slave met het eerste bericht in het verzendbuffer; indien deze berichten met elkaar overeenkomen, wordt het bericht uit het verzendbuffer verwijderd en wordt het antwoord doorgegeven naar de applicatie. In het geval dat het ontvan-

gen bericht niet overeen komt met het eerste bericht in het verzendbuffer, wordt het binnenkomende bericht genegeerd en zal de master de verbinding opnieuw synchroniseren.

In het geval dat de slave antwoordt met een errorbericht <ERR_ILLEGAL> dwz. de slave heeft een ongeldig bericht ontvangen, zal de master de verbinding tussen de master en slave opnieuw synchroniseren.

Indien de slave antwoordt met een errorbericht (anders dan ERR_ILLEGAL), wordt het bericht uit het verzendbuffer gehaald en de foutcode wordt doorgegeven aan de applicatie.

Time-out
In uitzonderlijke gevallen kan het voorkomen dat de master geen antwoord ontvangt, terwijl de onderliggende netwerklagen aangeven dat de verbinding tussen de master en slave goed is. Voor het onderkennen van deze situatie heeft de IVERA-master een timeout-timer. Na het verstrijken van de time-out zal de master de verbinding tussen de master en slave opnieuw synchroniseren.

Verbinding verbroken
Bij het wegvallen van de verbinding tussen master en slave, zal de onderliggende netwerklaag dit melden aan het IVERA-protocol. Bij het wegvallen van de verbinding zal het IVERA de lopende actie afbreken. Dit houdt in:

- de master verstuurt geen berichten meer totdat de verbinding opnieuw is opgebouwd.
- de slave stopt met versturen van een mogelijk antwoordbericht.

PING
In iedere IVERA-slave is een object “PING” aanwezig. Dit object heeft geen betekenis voor de slave, maar kan door de master gebruikt worden voor het synchroniseren van de master en slave. Tevens biedt PING de mogelijkheid tot het meten van de tijd dat een bericht onderweg is van master naar slave en terug naar de master.

Voor het synchroniseren van de master en slave schrijft de master een getal naar het object.

Master	Slave
@1#PING/#0=5	@1#:A

Tabel 3.25. Voorbeeld PING

3.9.6 Gebeurtenis in de slave (trigger)

Een slave kan de master informeren dat een event heeft plaatsgevonden, door het versturen van het bericht <BerichtSlaveTrigger>. De TriggerCode in het bericht is een applicatie specifieke code waarin kan worden gespecificeerd welk type event heeft plaatsgevonden.

De slave zal eerst een verbinding met de master opbouwen en vervolgens het bericht versturen. Het is vervolgens aan de master om de slave te ondervragen en de verbinding te verbreken. In het geval dat de master niet binnen een tijd van 5 minuten inlogt in de slave (via login op IVERA niveau) zal de slave automatisch de verbinding verbreken. Vanaf het moment dat door de master is ingelogd, geldt de normale login time-out.

N.B. Het <BerichtSlaveTrigger> is een uitzondering op het master-slave principe in die zin dat het bericht autonoom door de slave naar de master kan worden gestuurd.

3.10 Gereserveerde objectnamen

Voor het gebruik binnen het IVERA-protocol zijn een aantal objecten gereserveerd.

Object	Omschrijving
PING	Voor synchronisatie van master en slave.
LOGIN	Voor gebruikers-login.
BB0 t/m BB99	Base-object met per objecttype een lijst van objectnamen.
BBA0 t/m BBA99	Base-object met per objecttype een lijst van objectnamen inclusief alle objectattributen.
TID	Uniek toepassingidentificatienummer.
XID	Uniek automaatidentificatienummer
YID	Uniek applicatie-identificatienummer
ZID	Gereserveerd identificatienummer
AUTHOG	Gebruikersnamen
AUTHOP	Passwords

Tabel 3.26. Gereserveerde objectnamen

3.11 Verplichte objecten

Voor IVERA-compatibiliteit dienen in een slave minimaal de volgende objecten aanwezig te zijn:

- PING
- LOGIN
- BB0
- BB1
- BBA0
- BBA1
- TID
- XID
- YID
- AUTHOG
- AUTHOP

N.B. Voor IVERA VRI's geldt er een lijst met objecten die verplicht ondersteund dienen te worden. Zie paragraaf 2.8 van de IVERA objectdefinitie.

4. Beheer van objecten

De doelstelling van het IVERA-protocol is een fabrikantonafhankelijke oplossing voor de communicatie tussen een beheerscentrale en een VRI. De hiervoor gekozen oplossing is een eenvoudig protocol voor het lezen en schrijven van objecten. Echter het protocol op zich biedt geen enkele functionaliteit. De werkelijke functionaliteit ligt opgeslagen in de objecten. Bij het definieren van de objecten van een VRI spelen de volgende aspecten een rol:

- De functionaliteit is nog niet uitgekristalliseerd, waardoor er in de toekomst objecten zullen vervallen en andere zullen worden toegevoegd.
- Naast de standaard (basis) functionaliteit is er de mogelijkheid om per type automaat specifieke objecten te definiëren.
- Naast de standaardfunctionaliteit bestaat er de mogelijkheid om voor iedere (verkeerskundige-) applicatie specifieke objecten te definiëren.
- Welke objecten zijn in een VRI beschikbaar, dat wil zeggen; welke functionaliteit is er aanwezig in een VRI?
- De doelstelling is een zo eenvoudig, robuust en toekomst vast mogelijk protocol.

Vanwege bovenstaande redenen is er gekozen voor een identificatie van objecten op basis van een unieke naam.

Voor de naamgeving gelden de volgende randvoorwaarden:

- Een eenmaal uitgegeven naam mag worden verwijderd, maar mag **nooit** worden hergebruikt.
- De namen zijn onderverdeeld in 4 categorieën:
    - Algemene objecten per toepassing (A t/m W)
    - Automaatspecifieke objecten (X…).
    - Applicatiespecifieke objecten (Y…)
    - Gereserveerd (Z…)
- De algemene objecten per toepassing worden beheerd door een daarvoor aangewezen instantie die regelmatig vergadert en nieuwe namen uitgeeft.
- Bij het vrijgeven van nieuwe namen wordt een document meegeleverd waarin op formele wijze de volledige functionaliteit van het nieuwe object wordt beschreven.
- Het wijzigen van de functionaliteit van een object mag alleen plaatsvinden onder de voorwaarde van “upwards compatible”. Indien dit niet mogelijk is, dient een nieuw object te worden gedefinieerd met de gewenste functionaliteit.

4.1 Homoniemen

Tussen de verschillende toepassingen van het IVERA-protocol kunnen homoniemen ontstaan, d.w.z. een objectnaam komt in meerdere toepassingen voor en de functionaliteit van het object kan per toepassing verschillen. Om objecten uniek te identificeren is er een unieke toepassing identificatie (TID) die via het protocol uit de IVERA-slave kan worden gelezen.

Per toepassing worden de namen van automaat- en applicatiespecifieke objecten (alle objecten die beginnen met X, Y en Z) door de leverancier bepaald, hierdoor kunnen ook per toepassing homoniemen ontstaan. De functionaliteit van de objecten die beginnen met een X,Y of Z wordt bepaald door de naam in combinatie met een unieke identificatie (XID, YID, ZID). Deze identificaties zijn als objecten in de IVERA-slave opgeslagen.

Een voorbeeld van een homoniem is het object STATUS. Per toepassing bevat dit object de status van het aangesloten apparaat. Echter per toepassing kan betekenis van de status verschillen.

4.2 Algemene objecten

Binnen het IVERA-protocol is een aantal objecten gereserveerd. Deze objecten hebben een speciale betekenis binnen het protocol. Zie tabel 3.25. voor een overzicht van deze objecten.

4.3 Toepassingspecifieke objecten

Alle objecten die beginnen met de letters A t/m W zijn toepassingspecifiek. Dwz. de functionaliteit van deze objecten is eenduidig vastgelegd in combinatie met een unieke toepassingidentificatie (TID).

Voor de toepassingidentificatie (TID) zijn de volgende reserveringen gemaakt:

Bereik	Toepassing
1..9999	Verkeersregelinstallaties

Tabel 4.1. TID reserveringen

4.4 Automaatspecifieke objecten

Voor de automaatspecifieke objecten is de beginletter ‘X’ gereserveerd. De namen van de automaatspecifieke objecten worden door de leverancier van de automaat bepaald.

De functie van een automaatspecifiek object is eenduidig vastgelegd in combinatie met een unieke automaatidentificatie (XID).

4.5 Applicatiespecifieke objecten

Voor de applicatiespecifieke objecten is de beginletter ‘Y’ gereserveerd. De namen van de applicatiespecifieke objecten worden door de leverancier van de applicatie bepaald.

De functie van een applicatiespecifiek object is eenduidig vastgelegd in combinatie met een unieke applicatie-identificatie (YID).

5. Beveiliging en gebruikers

Deze paragraaf beschrijft de aanvulling op de IVERA-specificatie voor de beveiliging van een IVERA-toestel tegen inbreuk door derden.

5.1 Gebruikers

Het regeltoestel ondersteunt ten minste vier gebruikers. Hoewel het mogelijk is per regeltoestel meerdere gebruikers te definiëren, is het vanuit beheersoogpunt gewenst dit aantal laag te houden.

De opdrachtgever geeft bij opdracht de gebruikersnamen en passwords door aan de fabrikant. Indien niet opgegeven worden de gebruikersnamen en passwords door de fabrikant bepaald en bij levering schriftelijk aan de opdrachtgever gemeld.

5.2 Lokale wijziging van gebruikersnaam en password

De gebruikersnamen en passwords kunnen lokaal via een instelapparaat van het regeltoestel worden gewijzigd.

5.3 Wijziging gebruikersnaam en password

De gebruikers en bijbehorende passwords zijn gedefinieerd in de IVERA objecten:

- AUTHOG : Gebruikersnamen
- AUTHOP : Passwords

Het aantal elementen van deze objecten bepaalt het aantal gedefinieerde gebruikers.

Een lijst met gebruikersnamen kan worden opgevraagd door het lezen van het object AUTHOG. Door het schrijven naar het object AUTHOG kan een gebruikersnaam worden gewijzigd.
Het object AUTHOP levert bij lezen een lege string, door schrijven kan het password per gebruiker worden gewijzigd. Via het IVERA-protocol kunnen dus op afstand zowel gebruikersnaam als passwords worden gewijzigd.
Let wel: Iedere gebruiker die is ingelogd, kan deze actie uitvoeren. Om te voorkomen dat iemand zomaar een gebruikersnaam of password wijzigt, zal ook het huidige password moeten worden opgegeven.

Het wijzigen van een gebruikersnaam gebeurt door het schrijven naar het bijbehorende element van het object AUTHOG.

AUTHOG/<element>=”<gebruikersnaam>,<password>,<nieuw1>,<nieuw2>”

waarbij:

<element>	: object element (c.q. gebruiker)
<gebruikersnaam>	: huidige gebruikersnaam voor deze gebruiker.
<password>	: huidige password voor deze gebruiker of password van 1ste gebruiker.
<nieuw1>	: nieuwe gebruikersnaam voor deze gebruiker.
<nieuw2>	: moet overeenkomen met <nieuw1>.

Het wijzigen van een password gebeurt door het schrijven naar het bijbehorende element van het object AUTHOP.

AUTHOP/<element>=”<gebruikersnaam>,<password>,<nieuw1>,<nieuw2>”

waarbij:

<element>	: object element (c.q. gebruiker)
<gebruikersnaam>	: huidige gebruikersnaam voor deze gebruiker.
<password>	: huidige password voor deze gebruiker of password van 1ste gebruiker.
<nieuw1>	: nieuw password voor deze gebruiker.
<nieuw2>	: moet overeenkomen met <nieuw1>.

☞Door deze opzet heeft de 1ste gebruiker meer rechten. Namelijk de 1ste gebruiker kan de gebruikersnamen en passwords van alle gebruikers wijzigen. De overige gebruikers kunnen alleen hun eigen gebruikersnaam en password wijzigen.

☞Als alternatief voor het wijzigen van gebruikersnamen en toegangscode is een fabrikant specifieke methode toegestaan mits deze gebruik maakt van standaard FTP functionaliteit, d.w.z. het laden van een file in de slave. In dit geval dient de slave te beschikken over FTP server functionaliteit.

NB: De gebruikersnaam en password lengte zijn op MAX_FLEN gemaximaliseerd.

5.4 Beheersaspecten

Voor de beheerder heeft de IVERA beveiliging de volgende consequenties.

- Per nieuwe automaat de gebruikersnamen en passwords opgeven aan de fabrikant.
- Het beheren/wijzigen van de gebruikersnamen en passwords.

Gebruikersnamen

Voor de beheerder zijn er meerdere gebruikersnamen en passwords om toegang te verkrijgen tot het regeltoestel. De gebruikersnamen en passwords kunnen zowel lokaal als vanuit de centrale worden gewijzigd. Door de definitie van meerdere gebruikers kunnen verschillende partijen (zonder elkaars password te kennen) toegang verkrijgen tot het regeltoestel.

Voor de keuze van de gebruikersnamen zijn er meerdere mogelijkheden, maar er is een sterke voorkeur de gebruikersnamen eenvoudig te houden; twee mogelijkheden:

1. De gebruikersnaam voor alle regeltoestellen in het beheersgebied is identiek, bijvoorbeeld de naam van de beheersorganisatie.
2. De gebruikersnaam is uniek per regeltoestel, bijvoorbeeld een vast voorvoegsel gevolgd door de identificatie van het regeltoestel.

Andere mogelijkheden zoals volledig willekeurige namen per regeltoestellen zijn uiteraard mogelijk maar worden vanuit beheersoogpunt afgeraden.

Inbellen vanuit de IVERA centrale

De gebruikersnaam en het password van ieder regeltoestel zijn geconfigureerd in een IVERA-centrale. Een gebruiker die is ingelogd in de IVERA-centrale, heeft zodoende direct toegang tot alle regeltoestellen in het beheersgebied zonder gebruikersnaam en password te hoeven opgeven.

☞In principe zijn de gebruikersnamen en passwords van de regeltoestellen alleen bekend bij de ‘beheerder’. Een gemiddelde gebruiker van een IVERA-centrale heeft deze kennis niet.

Het wijzigen/beheren van de passwords

De gebruikersnamen en passwords worden per regeltoestel eenmalig geconfigureerd en kunnen tijdens de levensduur worden gewijzigd. Daar de gebruikersnamen relatief eenvoudig zijn, mag worden aangenomen dat de gebruikersnamen na enige tijd bekend zijn. De beveiliging is daarom voor 99% afhankelijk van het password. Het password van alle regeltoestellen zal regelmatig moeten worden gewijzigd. Het password zal voldoende ingewikkeld moeten zijn. De redenen om het password van een gebruiker of alle passwords te wijzigen zijn:

- Periodiek (1x per jaar).

- Na een inbreuk in het systeem van buitenaf.
- Bij een probleem in de beheersorganisatie waardoor de passwords buiten de organisatie bekend zijn geworden.
- Het feit dat iemand van buiten de organisatie “tijdelijk” toegang heeft gekregen tot het regeltoestel via een daarvoor vrijgegeven gebruikersnaam en password en deze toegang niet langer wenselijk is.

Voor de passwords zijn er meerdere mogelijkheden:

1. Één password voor alle regeltoestellen.
2. Een uniek password per regeltoestel.
3. Een uniek password per gebruiker per regeltoestel.

Methode 2 is in principe veiliger, omdat iemand van buitenaf die toegang tot 1 regeltoestel heeft verkregen niet direct toegang heeft tot alle regeltoestellen. Bij methode 1 is het van belang het password periodiek te wijzigen.

N.B. De gebruikersnamen en passwords die door de fabrikant eenmalig worden (voor)geprogrammeerd worden door de fabrikant niet geadministreerd. Als zodanig kan er op de fabrikant geen beroep worden gedaan, indien het password is vergeten/kwijtgeraakt. Wel kan de fabrikant de service verlenen om de gebruikersnamen en passwords (lokaal) opnieuw in te stellen.

De mate van beveiliging
Beveiliging is altijd een samenspel van een technische realisatie en de verantwoordelijke organisatie.

In eerste instantie is de IVERA-beveiliging bedoeld tegen inbreuk in het regeltoestel van buitenaf. Deze beveiliging is noodzakelijk omdat door het gebruik van gestandaardiseerde ‘open’ protocol iedereen met een internet-PC met het regeltoestel kan communiceren. De CHAP implementatie biedt de beveiliging door het gebruik van gebruikersnamen en passwords.

In tweede instantie is de IVERA-beveiliging bedoeld als bescherming binnen de eigen organisatie. Dit is mogelijk door de passwords alleen bekend te maken aan een beperkt aantal mensen binnen de organisatie die met de regeltoestellen moeten werken. Tevens is het mogelijk om aan verschillende partijen verschillende gebruikersnamen en passwords toe te kennen. In principe moet bij een wijziging in de samenstelling van de groep de passwords worden gewijzigd.

Alles tezamen biedt de CHAP implementatie een afdoende beveiliging die past bij de toepassing en de beheersorganisatie. Belangrijk is wel de beveiligingsgegevens binnen een zo select mogelijk gezelschap te houden en regelmatig de passwords te wijzigen.

“Tijdelijke” toegang tot het regeltoestel kan worden verschaft aan derden buiten de eigen organisatie door hiervoor een gebruikersnaam en password vrij te geven. Op het moment dat deze toegang niet meer wenselijk is kan de 1ste gebruiker de gebruikersnaam en/of password wijzigen en daarmee de toegang blokkeren. Zodoende is het zaak dat het password van de 1ste gebruiker te allen tijde binnen de eigen organisatie blijft.

6. Bijlage: BNF-notatie

In deze beschrijving van het IVERA-protocol wordt gebruik gemaakt van de Backus-Naur form (BNF) voor de beschrijving van het protocol. Hieronder volgt in het kort een uiteenzetting van deze notatie.

De beschrijving van het applicatieprotocol is opgebouwd uit zogenaamde BNF regels. Een BNF regel heeft het volgende formaat:

N = E

Waarbij N de naam is van een syntactische eenheid en E is een syntax expressie.

Een syntax expressie E heeft het formaat: T1 | T2 | .. | Tn

T1, T2, ..., Tn zijn syntax termen van E.

Een syntax term T heeft het formaat: F1 F2 ... Fn

F1, F2, ..., Fn zijn de syntax factoren van T. Een term definieert regels, waarbij een regel bestaat uit een regel met het formaat F1, gevolgd door een regel met het formaat F2, ... , gevolgd door een regel met het formaat Fn.

Een syntax factor F met het formaat: [E]

beschrijft een regel die leeg is of een regel met het formaat E (waarbij E een syntax expressie is).

Een syntax factor F met het formaat: { E }

beschrijft een regel die bestaat uit nul of meer regels met het formaat E (waarbij E een syntax expressie is).

Een syntax factor F met het formaat

N

refereert naar een syntax regel genaamd N.

een syntax factor F met het formaat

“ab..z”

definieert de symbolen ab .. z

NB. De beschrijving is overgenomen uit “Brinch Hanssen on Pascal compilers”.

7. Bijlage: Implementatieaspecten

Bij de implementatie van de IVERA-master en slave dient met de volgende aspecten rekening gehouden te worden.

- Het opbouwen en in stand houden van de verbinding valt buiten het IVERA-protocol.
- Een bericht is theoretisch onbeperkt in lengte.
- De prioriteit van de IVERA-slave in een VRI dient zodanig te worden ingesteld dat het regelprogramma geen last ondervindt van eventuele communicatie.
- De reactie tijd op een vraag vanuit de IVERA-master moet zodanig zijn dat real-time monitoring mogelijk is.
- De data binnen 1 antwoord bericht dient consistent te zijn (snapshot).

Daar de bovenstaande aspecten met elkaar in conflict zijn wordt volgende prioritering gebruikt:

1. Het regelprogramma dient onder alle omstandigheden goed te blijven functioneren.
2. De informatie die uit de IVERA-slave wordt gelezen dient consistent te zijn.
3. Reactie snelheid.
4. Data throughput.

8. Bijlage: IVERA objectbeschrijving

IVERA Objectbeschrijving

Naam:	(max. 16 karakters)	
Omschrijving:	(max. 32 karakters)	
Type:	(0 = getal, 1 = tekst)	

		Groep	1	2	3	4
Gebruikersrechten:	(4=lezen, 6=lezen/schrijven)					

Logboek:	(Ja/Nee)	

	Dimensie	1	2	3
Aantal data-elementen:				
Index data-elementen:				

Minimumwaarde data-element waarde:	
Maximumwaarde data-element waarde:	

Index data minimum elementwaarde:	
Index data maximum elementwaarde:	

Data-element formaat:	
Data-element stapgrootte:	

Functie van het object:

Toepassingsidentificatienummer(s) (TID):		
Automaatidentificatienummer(s) (XID):		
Applicatie-identificatienummer(s) (YID):		