De CVN RIS Interface

Beschrijving van
de software-interface voor
RIS-informatie tussen
het applicatieprogramma en
de procesbesturing
voor verkeersregeltoestellen

CVN Commissie C
Auteurs : Peter Goossens, Ton van Grinsven, Peter Smit, Jaap van Boxtel,
Raymond Cuenen, Marcel Fick, Patrick Huijskes
versienummer: ~~1.2~~ 2.0
datum: ~~10 juni 2020~~12 juni 2022
status: ~~vastgesteld door CVN~~vastgesteld door CVN

Inhoud

1. Inleiding

Dit document beschrijft de CVN RIS Interface.

Het doel van de CVN RIS Interface, is het standaardiseren van de RIS-FI ontsluiting in een CCOL applicatie. Het gevolg hiervan is dat CCOL applicaties die gebruik maken van RIS functionaliteit, zonder aanpassing uitgewisseld kunnen worden tussen fabrikanten van ITS applicatie containers (lees: de verschillende hardware boxen waarin een ITS applicatie draait).

Er is gekozen om de interface op te bouwen uit twee op elkaar aansluitende abstractielagen. Dit zijn respectievelijk de communicatie laag en de presentatie laag uit het OSI model voor datacommunicatie.

De communicatie laag is een abstractie van de RIS-FI uit de iVRI architectuur, die het asynchrone karakter van deze interface behoudt. Hiermee kan de CCOL-programmeur gemakkelijk de events van de RIS afhandelen op het moment dat deze optreden op een object-georiënteerde manier.

De presentatie laag is gebaseerd op een representatie van de RIS-FI in databuffers zoals te doen gebruikelijk in de CVN-C interface. Deze interface sluit aan op de huidige common practices in de CCOL community.

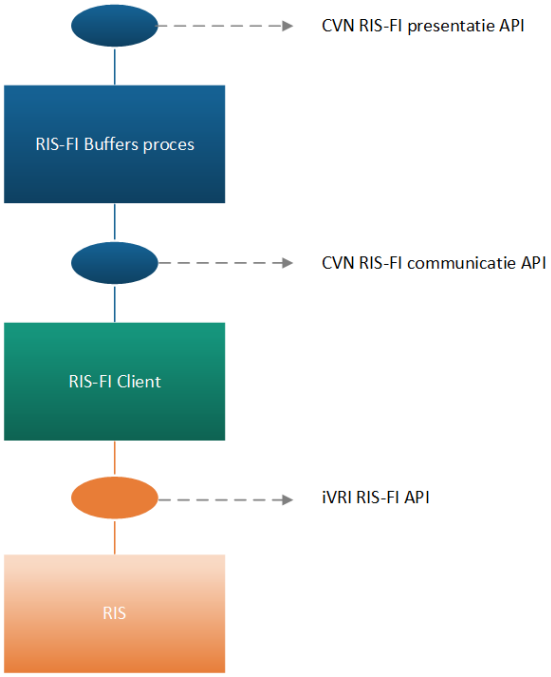
Door deze gelaagde opbouw biedt de interface zowel de ruwe kracht van de RIS-FI als de convenience van de CCOL databuffers.

Het is aan de gebruiker om te kiezen op welke laag hij zijn applicatie wil gaan realiseren, afhankelijk van zijn behoefte.

Die keuze zal gebaseerd zijn op de behoefte van de CCOL gebruiker. We zien hier grofweg twee type gebruikers:

- Gebruikers die op de gebruikelijke wijze gebruik willen maken van databuffers.
- Gebruikers die gebruik willen maken van meer geavanceerde wijze van benaderen van RIS objecten, object oriëntatie en gebruik willen maken van het asynchrone karakter van de RIS-FI.

Dit document is tot stand gekomen naar aanleiding van de uitdrukkelijke wens van de CVN om, daar waar CCOL wordt gebruikt binnen ITS-applicaties, de interface tussen CCOL en de RIS-FI te standaardiseren. Met als doel de CCOL applicaties, die in deze context ontstaan, uitwisselbaar en fabrikant onafhankelijk te laten zijn.



Figuur 1 De verschillende gestandardiseerde API's

2. Waarborgen van compatibiliteit

Om de compatibiliteit van CCOL applicaties over verschillende containers te waarborgen gelden de volgende spelregels:

- Elke container ondersteunt verplicht zowel de communicatie laag als de presentatie laag.
- At run time is slechts 1 van beide type interfaces actief (om mogelijke conflicten bij het schrijven naar de interface te voorkomen).

3. Scope

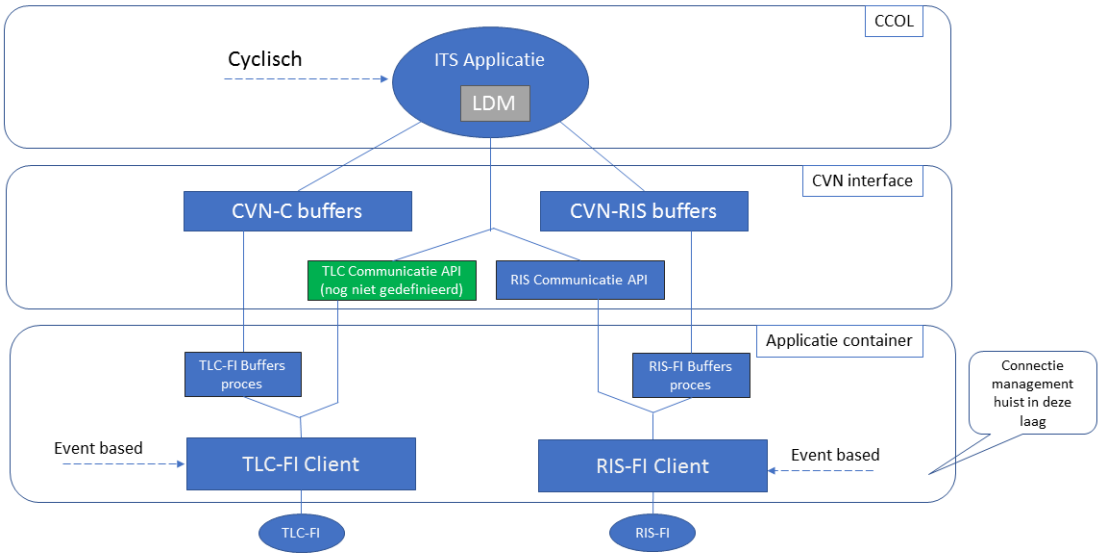
De Scope van de in dit document beschreven interface beslaat het CVN interface vak in onderstaande figuur.

De TLC-FI ontsluiting wordt niet besproken in dit document omdat dit onderdeel al is gestandaardiseerd in de bekende CVN-C interface.

De realisatie van een LDM valt buiten deze standaardisatie opgave en bevindt zich bij gevolg in de ITS applicatie (zie onderstaande figuur).

De leveranciers van de applicatiecontainers zijn verantwoordelijk voor de implementatie van de softwareprocessen die de CVN RIS interface realiseren (in het vak 'Applicatie container').

Toekomstvisie: Merk op dat het voor de hand ligt om de TLC-FI ontsluiting op gelijke wijze te gaan ontsluiten als de RIS-FI. Dat betekent dan dat het groene blokje in figuur 2, dat nu nog niet gedefinieerd is, gestandaardiseerd en aangeboden dient te worden. Dit onderdeel maakt echter geen onderdeel uit van deze standaardisatie opgave.



Figuur 2 Een overzicht van de ITS applicatie architectuur

4. CVN RIS-FI presentatie API

Inleiding
Het document iVRI2_del_1b_IDD_RIS-FI_v1.2 omschrijft alle RIS objecten. Het object ItsStation dekt nagenoeg de gehele scope met de bijbehorende CAM berichten en wordt aangevuld met prioriteringsberichten voor openbaar vervoer en hulpdiensten (SRM- en SSM berichten). De overige objecten hebben voor nu minder toegevoegde regeltechnische meerwaarde. Het koppelvlak heeft tot doel om de relevante informatie binnen te halen in de CCOL-applicatie. Het binnen de CCOL-applicatie gevulde buffer wordt voor diverse doelen toegepast. Te denken aan het ophalen van het aantal voertuigen op een opgegeven richting binnen een afstand gebied of het detecteren van vrachtverkeer ten behoeve van tovergroen.

Variabelen/woordlengte
Alle variabelen en constanten in de RIS interface hebben, voor een betere herkenbaarheid, een naam gekregen die altijd begint met het voorvoegsel "rif_", "Rif_" of "RIF_" (RIF staat voor **R**IS **I**nter**F**ace). De naamgevingen zijn verder zoveel mogelijk overeenkomstig met de naam-gevingen van de objecten omschreven in de RIS-FI specificatie.
Voor de variabelen, die in de RIS interface worden vastgelegd, worden aparte typen variabelen gedefinieerd (typedef).
Daar waar een relatie van variabelen binnen het applicatieprogramma met de interface variabelen aanwezig is moeten ook deze variabelen herleid zijn naar de in de interface benoemde type variabelen. De woordlengtes van die variabelen zijn op die manier eenvoudig aan te passen bij implementatie van de applicatie in het regeltoestel.

In de RIS interface (RIF) worden de volgende typen variabelen gebruikt:
- `rif_int`
- `rif_float`
- `rif_bool`
- `rif_timestamp`
- `rif_string`

De fabrikant dient bij implementatie er voor zorg te dragen dat bij de typedefine de juiste type variabelen worden gekozen.

Interface
Het koppelvlak zoals beschreven in Appendi is van toepassing, naamgevingen zijn zoveel mogelijk overeenkomstig gehouden met de objecten omschreven in de RIS-FI specificatie. Aanvullend is de variabele 'valid' opgenomen om vast te stellen of het om een geldig voertuig gaat. Een niet geldig voertuig is afgemeld en zal uit het buffer worden verwijderd. Het koppelvlak wijkt qua opbouw af ten opzichte van bijvoorbeeld het DSI-buffer. Het DSI-buffer omvat slechts een melding om door de applicatie over te nemen. Het RIS-buffer kan per lees- en schrijf actie meer dan 1 bericht omvatten.

In het bestand 'ris_defs.h' zijn de RIS objecten gedefinieerd:
- `struct Rif_ItsStation (CAM)`
- `struct Rif_PrioritizationRequest (SRM)`
- `struct Rif_ActivePrioritization (SSM)`
- `struct Rif_ItsEvent (DENM)`

- struct Rif_SignalGroup;
- struct Rif_SignalHead;
- struct Rif_ProductInformation;

In het bestand 'rif.inc' zijn de databuffers voor de afhandeling van de RIS berichten gedefinieerd. Per bericht type wordt een array van RIS objecten gedefinieerd met een leesaanwijzer en een schrijfaanwijzer.

```
/* ITSSTATION_PB[]-buffer (CAM) */
/* ---------------------------- */
/* De buffer waar alle ITS stations in worden gezet. */
/* Deze wordt geschreven door de PB en gelezen door de AP. */

   struct Rif_ItsStation RIF_ITSSTATION_PB[RIF_MAX_ITSSTATION_PB];

   rif_int RIF_ITSSTATION_PB_WRITE;  /* Te schrijven door de PB */
   rif_int RIF_ITSSTATION_PB_READ;   /* Te schrijven door de AP */

/* PRIOREQUEST_PB[]-buffer (SRM) */
/* ----------------------------- */
/* De buffer waar alle prioritization requests in worden gezet. */
/* Deze wordt gelezen door de AP en geschreven door de PB. */

   struct Rif_PrioritizationRequest RIF_PRIOREQUEST_PB[RIF_MAX_PRIOREQUEST_PB];

   rif_int RIF_PRIOREQUEST_PB_WRITE; /* Te schrijven door de PB */
   rif_int RIF_PRIOREQUEST_PB_READ;  /* Te schrijven door de AP */

/* ACTIVEPRIO_AP[]-buffer (SSM) */
/* ---------------------------- */
/* De buffer waar alle active prioritizations in worden gezet. */
/* Deze wordt geschreven door de AP en gelezen door de PB. */

   struct Rif_ActivePrioritization RIF_ACTIVEPRIO_AP[RIF_MAX_ACTIVEPRIO_AP];

   rif_int RIF_ACTIVEPRIO_AP_READ;  /* Te schrijven door de PB */
   rif_int RIF_ACTIVEPRIO_AP_WRITE; /* Te schrijven door de AP */

/* ITSEVENT_AP[]-buffer (DENM) */
/* --------------------------- */
/* De buffer waar alle ITS events vanuit de AP in worden gezet. */
/* Deze wordt geschreven door de AP en gelezen door de PB. */

   struct Rif_ItsEvent RIF_ITSEVENT_AP[RIF_MAX_ITSEVENT_AP];

   rif_int RIF_ITSEVENT_AP_READ;  /* Te schrijven door de PB */
   rif_int RIF_ITSEVENT_AP_WRITE; /* Te schrijven door de AP */
```

```
/* ITSEVENT_PB[]-buffer (DENM) */
/* --------------------------- */
/* De buffer waar alle ITS events vanuit de PB in worden gezet. */
/* Deze wordt geschreven door de PB en gelezen door de AP. */
/* De events die hier in komen zijn alleen de events van andere applicaties. */

   struct Rif_ItsEvent RIF_ITSEVENT_PB[RIF_MAX_ITSEVENT_PB];

   rif_int RIF_ITSEVENT_PB_WRITE; /* Te schrijven door de PB */
   rif_int RIF_ITSEVENT_PB_READ;  /* Te schrijven door de AP */
```

De volgende definities stellen de grootten van de databuffers in:

```
/* Definities grootte databuffers */
/* ------------------------------ */
   #define RIF_MAX_ITSSTATION_PB   100   /* CAM berichten procesbesturing    */
   #define RIF_MAX_PRIOREQUEST_PB   25   /* SRM berichten procesbesturing    */
   #define RIF_MAX_ACTIVEPRIO_AP    10   /* SSM berichten applicatie         */
   #define RIF_MAX_ITSEVENT_AP      10   /* DENM berichten applicatie        */
   #define RIF_MAX_ITSEVENT_PB      10   /* DENM berichten procesbesturing   */
```

In het bestand 'rif.inc' is ~~ook~~ een variabele gedefinieerd voor de doorgifte van de actuele tijd (UTC-tijd) vanuit de procesbesturing.

```
/* RIF_UTC_TIME_PB */
/* --------------- */
/* Timestamp RIF_UTC_TIME_PB is de actuele tijd. */
/* Het aantal milliseconden sinds 1-1-1970 00:00:00:00 UTC. */
/* Deze UTC-tijd wordt geschreven door de PB en gelezen door de AP. */

   rif_timestamp RIF_UTC_TIME_PB;   /* Te schrijven door de PB */
```

In het bestand 'rif.inc' zijn variabelen gedefinieerd voor de doorgifte van de snelheidsprofielen, wachtrijlengten, type verkeerslantaarns en productinformatie vanuit de applicatie.

```
/* RIF_SIGNALGROUP_AP[]-buffer */
/* --------------------------- */
/* RIF_SIGNALGROUP_AP[] bevat de snelheidsprofielen (type, distance, speed) en    */
/* wachtrijlengten (zoneLength, lane) van de signaalgroepen van de ITS-applicatie.*/
/* Met de wijzigingsvlag RIF_SIGNALGROUP_AP_SPEEDPROFILE_CHNG[] wordt aangegeven  */
/* dat het snelheidsprofiel van een signaalgroep is gewijzigd.                    */
/* Met de wijzigingsvlag RIF_SIGNALGROUP_AP_QUEUELENGTH_CHNG[] wordt aangegeven   */
/* dat de wachtrijlengte van een signaalgroep is gewijzigd.                       */

   struct Rif_SignalGroup RIF_SIGNALGROUP_AP[FCMAX];

   rif_int RIF_SIGNALGROUP_AP_SPEEDPROFILE_CHNG[FCMAX]; /* SpeedProfile gewijzigd */
   rif_int RIF_SIGNALGROUP_AP_QUEUELENGTH_CHNG[FCMAX];  /* QueueLength gewijzigd  */
```

```
/* RIF_SIGNALHEAD_AP[]-buffer */
/* -------------------------- */
/* RIF_SIGNALHEAD_AP[] bevat het type verkeerslantaarn (type, mainLightIndex)     */
/* van de signaalgroepen van de ITS-applicatie.                                   */
/* het type verkeerslantaarn is van belang voor de bijzondere lichten.            */
/* RIF_SIGNALHEAD_AP[] wordt geschreven door de AP en gelezen door de PB.         */

   struct Rif SignalHead RIF_SIGNALHEAD_AP[FCMAX];


/* RIF_ITSINFO_AP */
/* -------------- */
/* RIF_ITSINFO_AP bevat de productinformatie (manufacturerName, certifiedName,    */
/* certifiedVersion, version) van de ITS-Applicatie.                              */
/* RIS_ITSINFO_AP wordt geschreven door de AP en gelezen door de PB.              */

   struct Rif ProductInformation RIF_ITSINFO_AP;
```

Lezen en schrijven tussen applicatie en procesbesturing
Vanuit de procesbesturing wordt de RIS-data aangeboden, zowel de CAM als SRM-berichten. Een lees- en schrijf aanwijzer geven aan tot waar is gelezen en tot waar is geschreven. Schrijven vindt plaats vanuit de procesbesturing. Lezen vindt alleen plaats vanuit de applicatie. De schrijf aanwijzer is beschikbaar om vanuit de applicatie te lezen (CAM-berichten). Deze mag niet vanuit de applicatie worden aangepast:

```
rif_int RIF_ITSSTATION_PB_WRITE;  /* Te schrijven door de PB */
rif_int RIF_ITSSTATION_PB_READ;   /* Te schrijven door de AP */
```

Per lees-actie wordt een geheel object gelezen vanuit de structure RIF_ITSSTATION_PB:

```
/* De buffer waar alle ITS stations in worden gezet. */
/* Deze wordt gelezen door de AP en geschreven door de PB. */
struct Rif_ItsStation RIF_ITSSTATION_PB[RIF_MAX_ITSSTATION_PB];
```

Bij het starten van de applicatie staan beide aanwijzers op 0. Zodra er informatie in het buffer wordt weggeschreven, zal de procesbesturing de schrijf-aanwijzer aanpassen. De applicatie zal per machineslag op een verschil in beide aanwijzers toetsen en de beschikbare CAM-data overnemen en in een applicatie-buffer wegschrijven. Een bestaand object wordt overschreven. Het lezen vindt plaats op de hoogte van de lees-aanwijzer. De lees-aanwijzer wordt elke leesslag opgehoogd met 1 stap na het lezen van de data. Met die beperking dat per machineslag een maximaal aantal elementen zal worden opgehaald. Bij het starten van de applicatie zullen in beide buffers de ObjectID's worden gewist. Een leeg ObjectID houdt in dat alle achterliggende data niet geldig is.
De buffers worden niet bewaakt op overschrijden. De grootte zal voorafgaand aan de implementatie vastgesteld worden. Wanneer de grootte wordt overschreden, zullen oudere berichten automatisch worden overschreven.

De structure van de databuffer aan de zijde van de CCOL-applicatie voor het bijhouden en bewaren van de CAM-informatie is bij voorkeur identiek. Dit om kopieer slagen eenvoudig te houden. Een geheel element kan op deze wijze worden bewerkt. Het voordeel hiervan is dat

de structure kan worden aangepast zonder dat dit impact heeft op de diverse lees- en schrijf acties.

Het bovengenoemde principe is eveneens beschikbaar voor de te ontvangen SRM-data.
Het lezen en schrijven van SRM-data vindt op gelijke wijze plaats en wel met de pointers:

```
rif_int RIF_PRIOREQUEST_PB_WRITE; /* Te schrijven door de PB */
rif_int RIF_PRIOREQUEST_PB_READ;  /* Te schrijven door de AP */
```

Per lees-actie wordt een geheel object gelezen vanuit de structure RIF_PRIOREQUEST_PB:

```
/* De buffer waar alle prioritization requests in worden gezet. */
/* Deze wordt gelezen door de AP en geschreven door de PB. */
struct Rif_PrioritizationRequest RIF_PRIOREQUEST_PB[RIF_MAX_PRIOREQUEST_PB];
```

Elk gelezen element wordt weggeschreven in een applicatie-buffer. Dit buffer wordt in de paragraaf ‘Inlezen CCOL-ITS applicatie’ beschreven.

Vanuit de applicatie wordt de RIS-data aangeboden van de SSM-berichten. Een lees- en schrijfaanwijzer geven aan tot waar is gelezen en tot waar is geschreven. Schrijven vindt plaats vanuit de applicatie. Lezen vindt alleen plaats vanuit de procesbesturing. De schrijf aanwijzer is beschikbaar om vanuit de procesbesturing te lezen (SSM-berichten). Deze mag niet vanuit de procesbesturing worden aangepast:

```
rif_int RIF_ACTIVEPRIO_AP_WRITE; /* Te schrijven door de AP */
rif_int RIF_ACTIVEPRIO_AP_READ;  /* Te schrijven door de PB */
```

Per lees-actie wordt een geheel object gelezen vanuit de structure RIF_ACTIVEPRIO_AP:

```
/* De buffer waar alle ITS station in worden gezet. */
/* Deze wordt geschreven door de AP en gelezen door de PB. */
struct Rif_ActivePrioritization RIF_ACTIVEPRIO_AP[RIF_MAX_ACTIVEPRIO_AP];
```

Vanuit de applicatie en de procesbesturing kan DENM-data worden aangeboden.
Het lezen en schrijven van DENM-data vanuit de applicatie vindt plaats met de pointers:

```
rif_int RIF_ITSEVENT_AP_READ;  /* Te schrijven door de PB */
rif_int RIF_ITSEVENT_AP_WRITE; /* Te schrijven door de AP */
```

Per lees-actie wordt een geheel object gelezen vanuit de structure RIF_ITSEVEN_AP:

```
/* De buffer waar alle ITS events vanuit de AP in worden gezet. */
/* Deze wordt geschreven door de AP en gelezen door de PB. */
struct Rif_ItsEvent RIF_ITSEVENT_AP[RIF_MAX_ITSEVENT_AP];
```

Het lezen en schrijven van DENM-data vanuit de procesbesturing vindt plaats met de pointers:

```
rif_int RIF_ITSEVENT_PB_WRITE; /* Te schrijven door de PB */
rif_int RIF_ITSEVENT_PB_READ;  /* Te schrijven door de AP */
```

Per lees-actie wordt een geheel object gelezen vanuit de structure RIF_ITSEVEN_PB:

```
/* De buffer waar alle ITS events vanuit de PB in worden gezet. */
/* Deze wordt geschreven door de PB en gelezen door de AP. */
struct Rif_ItsEvent RIF_ITSEVENT_PB[RIF_MAX_ITSEVENT_PB];
```

Initialisatie
Initialisatie van de RIS interface geschiedt door het applicatieprogramma. Voor het initialiseren worden twee verschillende situaties onderscheiden:
1. initialisatie na programmastart en
2. initialisatie voorafgaand aan de toestand regelen.

Ad 1.
Bij initialisatie na programmastart worden door het applicatieprogramma alle variabelen in de interface geïnitialiseerd.

Ad 2
Bij initialisatie voorafgaand aan de toestand regelen worden alleen die variabelen in de interface geïnitialiseerd die voor het inschakelen van de regeling van belang zijn.

Lezen en schrijven in de interface
Tijdens uitvoering van de functie applicatieprogramma() mag door de procesbesturing niet in de interface worden geschreven of gelezen. De procesbesturing dient ervoor te zorgen dat bij aanroep van de functie applicatieprogramma() alle informatie in de interface voor het applicatieprogramma beschikbaar is.

Inlezen CCOL-ITS applicatie
Binnen de CCOL applicatie dienen buffers te worden gedefinieerd voor het bewaren en bijhouden van de actuele CAM en SRM informatie, bijvoorbeeld:
```
struct Rif_ItsStation ITSSTATION_AP[RIF_MAX_ITSSTATION_AP];
struct Rif_PrioritizationRequest PRIOREQUEST_AP[RIF_MAX_PRIOREQUEST_AP];
```

De structure van deze buffers is identiek als aan de procesbesturingszijde. Deze buffers zullen tenminste even groot moeten zijn als de buffers aan de procesbesturingszijde. Bij voorkeur veel groter.
Elk ontvangen RIS-element zal worden getoetst op plausibiliteit:
1. Het ObjectID moet qua karakters voldoen `/* Allowed characters: ‘a-z’ (ASCII 97 through 122), ‘A-Z’ (ASCII 65 through 90), ‘0-9’, ‘_’ (underscore, ASCII 95) and ‘-‘ (hyphen, ASCII 45). */;`
2. Het kruispunt. Is deze niet gelijk aan het kruispuntnummer van deze CCOL-applicatie, zal er een controle plaatsvinden op een ObjectID van het kruispunt (`SYSTEM_ITS`). Bestaat deze laatste, zal deze voor gaan. Bestaat deze niet, zal het element worden opgenomen in het buffer en zal de afstand worden gecorrigeerd.
3. Wanneer valid gelijk is aan 0, wordt het element in het CCOL-buffer opgezocht en verwijderd.

Als gevolg van het verwijderen van elementen uit het buffer, kunnen gaten ontstaan. Deze gaten worden opgevuld, door het laatste element op deze plek te plaatsen. Het voordeel hiervan is, dat per uitleesslag slechts gelezen wordt tot het ObjectID leeg is. Dit spaart processortijd.
Daarnaast geldt dat bij een leeg ObjectID, de eventuele onderliggende data geen waarde heeft!

5. CVN RIS-FI communicatie API

Inleiding
Het document iVRI2_del_1b_IDD_RIS-FI_v1.2 beschrijft de RIS-Facilities Interface (RIS-FI) met de bijbehorende RIS-Objecten. Om een ITS-regelapplicatie te maken, op basis van CCOL, die gebruik maakt van de beschikbare RIS-data, zoals CAM berichten aangevuld met prioriteringsberichten voor openbaar vervoer en hulpdiensten (SRM- en SSM berichten), zal een implementatie van deze RIS-FI interface beschikbaar moeten zijn.
Aangezien de CCOL-applicatieprogrammeur echter alleen geïnteresseerd is in de relevante informatie zal de procesbesturing de verbinding met de RIS en de afhandeling van de voor de RIS-FI gedefinieerde objecten verzorgen en afschermen voor de CCOL-applicatie. Deze afscherming noemen we de *RIS Interface Façade* (oftewel RIF-Facade).
Het koppelvlak heeft als doel de relevante informatie uit te wisselen tussen de CCOL-applicatie en de RIS, waarbij het zogenaamde 'event-based' karakter van de RIS-FI behouden blijft. Hierdoor kan de CCOL-applicatie ook asynchroon databewerking doen ten behoeve van het regelen.

Interface
Als basis wordt uitgegaan van de objecten-definities zoals die in het vorige hoofdstuk zijn beschreven:

```
// rif_defs.h
#ifndef _RIF_DEFS_H
#define _RIF_DEFS_H

// Zie appendix voor verdere definitie
struct Rif_ItsStation;
struct Rif_PrioritizationRequest;
struct Rif_ActivePrioritization;
struct Rif_ItsEvent;
struct Rif_SignalGroup;
struct Rif_SignalHead;
struct Rif_ProductInformation;

#endif /* _RIF_DEFS_H */
```

De RIF-Facade zelf is een C++ template klasse, geïmplementeerd als singleton. Voor elk RIS-object uit 'rif_defs.h' is er een specialisatie klasse beschikbaar, die de afhandeling van de notificaties en het versturen van de objecten van dat bijbehorende type regelt.

```
// rif_facade.h
#ifndef _RIF_FACADE_H
#define _RIF_FACADE_H
```

```c
#include <stddef.h>
#include "rif_defs.h"

```

```
/* -------------------- */
/* RIS Interface Facade */
/* -------------------- */
/* Het '_Type' moet gelijk zijn aan een van de RIS-objecten. */

template<class _Type>
class Rif_Callback {
public:
    /* Definitie van de notificatie-functie. */
    /* Deze functie zal door de PB worden aangeroepen voor elk ontvangen RIS-object. */
    virtual void OnNotifyObject(const _Type& object) = 0;
};

template<class _Type>
class Rif_Facade {
public:
    /* Voor elk _Type is er één facade beschikbaar. */
    static Rif_Facade<_Type>& get() {
        static Rif_Facade<_Type> instance;
        return instance;
    }

    /* Methode voor het aanmelden voor notificaties van ontvangen RIS-objecten. */
    void subscribe(Rif_Callback<_Type>* callback);
    /* Methode voor het verzenden van RIS-objecten. */
    void publish(const _Type& object);

private:
    Rif_Facade() {}                    // Constructor is niet publiekelijk!
    Rif_Facade(Rif_Facade const&);     // Kopiëren is niet toegestaan!
    void operator=(Rif_Facade const&); // Toewijzen is niet toegestaan!
};

#endif /* _RIF_FACADE_H */
```

Merk op dat alleen de functies “subscribe” en “publish” een specialisatie hebben voor de relevante RIS-objecten. Deze implementatie is afhankelijk van de procesbesturing en de koppeling met de RIS-FI Client implementatie. Hierbij is de procesbesturing ook verantwoordelijk voor de asynchrone afhandeling van de aanroep van de ‘notificatie-functie’. De volgende specialisaties zijn voor de CCOL-applicatie gedefinieerd:

```
// rif_facade.cpp
#include "rif_facade.h"

template<>
void Rif_Facade<Rif_ItsStation>::subscribe(_Callback handler) {
   // Registreer 'handler' voor notificaties van ItsStations.
}

template<>
```

```
void Rif_Facade<Rif_PrioritizationRequest>::subscribe(_Callback handler) {
   // Registreer 'handler' voor notificaties van PrioritizationRequests.
}

template<>
void Rif_Facade<Rif_ItsEvent>::subscribe(_Callback handler) {
   // Registreer 'handler' voor notificaties van ItsEvents.
}

template<>
void Rif_Facade<Rif_ActivePrioritization>::publish(const Rif_ActivePrioritization&
active_prioritization) {
   // Verzend 'active_prioritization' naar de RIS.
}

template<>
void Rif_Facade<Rif_ItsEvent>::publish(const Rif_ItsEvent& its_event) {
   // Verzend 'its_event' naar de RIS.
}
```

Afhandeling

Ontvangen van notificaties

Vanuit de procesbesturing wordt de RIS-data aangeboden, zowel de CAM-, de DENM- als de SRM-berichten. Dit gebeurt op het moment dat deze data via de RIS-FI beschikbaar komt. De applicatie kan deze data ontvangen via de notificatie-functie van een opgegeven Rif_Callback implementatie. Deze functie moet eerst worden geregistreerd bij de desbetreffende Rif_Facade klasse door middel van de “subscribe” methode. Voor het ontvangen van ItsStation-objecten bijvoorbeeld:

```
void handle_cam(const Rif_ItsStation& its_station)
{
    // Voer bewerkingen uit
}

void init_application(void)
{
     /* RIF Initialisatie */
    Rif_Facade<Rif_ItsStation>::get().subscribe(
        FunctionCallback<Rif_ItsStation>::Wrap(handle_cam));
}
```

Het ontvangen van de PrioritizationRequest- en ItsEvent-objecten gaat op een vergelijkbare manier. Het ontvangen object is alleen geldig voor de duur van de notificatieaanroep. Wanneer deze aanroep beëindigd zal ook het object uit het geheugen worden verwijderd. Alle notificaties worden sequentieel aangeboden aan de notificatie-functie. Dit betekent dat de applicatie eerst het ontvangen object moet verwerken (functie-beëindiging) alvorens een volgend object wordt aangeboden.

De Rif_Facade klasse accepteert geen meerdere notificatie-functies tegelijk. Bij eventuele meerdere aanroepen van de “subscribe” methode zal alleen de laatste aangeboden functie gebruikt worden. Dit betekent dat, functioneel gezien, een afmelding uitgevoerd kan worden door de “subscribe” methode met de NULL pointer aan te roepen.

Verzenden van RIS-objecten

Vanuit de regelapplicatie kan RIS-data worden aangeboden dat zal resulteren in het verzenden van SSM- en/of DENM-berichten. Op elk gewenst moment kan de applicatie de “publish” methode van de desbetreffende Rif_Facade klasse aanroepen om informatie naar de RIS te sturen. Voor het verzenden van een antwoord op een prioriteringsbericht bijvoorbeeld:

```
void handle_srm(const Rif_PrioritizationRequest& request)
{
    Rif_ActivePrioritization response;
    strcpy(response.id, request.id);
    response.sequenceNumber = request.sequenceNumber;
    response.prioState = RIF_PRIORITIZATIONSTATE_PROCESSING;

    /* Verstuur antwoord */
    Rif_Facade<Rif_ActivePrioritization>::get().publish(response);
}
```

Verzenden van ItsEvent-objecten gaat op een vergelijkbare manier. Het aangeboden object is na de aanroep van de “publish” methode niet meer nodig voor de Rif_Facade klasse en kan dus uit het geheugen worden verwijderd.

De Rif_Facade klasse zal op aanroep van de “publish” methode het aangeboden object converteren en verzenden via de RIS-FI. Dit gebeurt zoveel mogelijk gelijktijdig, zodat de applicatie door kan gaan met de verdere verwerkingen voor de regeling.

6. Appendix: rif_defs.h

```
/* RIS INTERFACE: PROCESBESTURING - APPLICATIEPROGRAMMA */
/* ================================================== */

/* RIS  :  versie 1.22.0    */
/* FILE :  rif_defs.h    */
/* DATUM:  12-06-2022 - versie 2.0 - Consolidatie uitbreidingen + RIF_STRINGSIZE    */
/* DATUM:  10-06-2020 - versie 1.2 - toelichting bij eta; eta=0 onbekend (unknown)  */
/*                      RIF_VEHICLESUBROLE_PLATOON en RIF_VEHICLESUBROLE_ECODRIVING */
/* DATUM:  05-10-2019 - versie 1.1 - toelichting bij signalGroup aangescherpt;
                       signalGroup ID is zonder Intersection ID */
/* DATUM:  17-09-2018 - versie 1.0 - toelichting bij signalGroup aangepast */
/* DATUM:  12-09-2018 - versie 1.0   */

#ifndef _RIF_DEFS_H
#define _RIF_DEFS_H

/* include file */
/* ============ */
#include <stdint.h>

/* Typen variabelen */
/* ================ */
#define RIF_FALSE         0
#define RIF_TRUE          1
#define RIF_STRINGLENGTH 63
#define RIF_STRINGSIZE   64

typedef uint64_t rif_timestamp;   /* uint64_t defined in: stdint.h */
typedef int      rif_int;
typedef double   rif_float;
typedef short    rif_bool;
typedef char     rif_string[RIF_STRINGSIZELENGTH + 1];

/* Definitie array grootte */
/* ======================= */
#define RIF_MAXLANES  4    /* grootte array matches[RIF_MAXLANES] voor CAM berichten  */
#define RIF_MAXTRACES 5    /* grootte array traces[RIF_MAXTRACES] voor DENM berichten */
#define RIF_MAXPOINTS 10   /* grootte array points[RIF_MAXPOINTS] voor DENM berichten */

#define RIF_MAXQUEUES             16   /* grootte array queue[]                    */
#define RIF_MAXSPEEDPROFILES      16   /* grootte array speedProfile[]             */
#define RIF_MAXSIGNALGROUPSINCAM  8    /* grootte array signalGroup[] in CAM       */




/* Type definities */
/* =============== */

/* StationType */
/* ----------- */
enum Rif_StationType {
    RIF_STATIONTYPE_UNKNOWN         =  0,
```

```
    RIF_STATIONTYPE_PEDESTRIAN      =  1,
    RIF_STATIONTYPE_CYCLIST         =  2,
    RIF_STATIONTYPE_MOPED           =  3,
    RIF_STATIONTYPE_MOTORCYCLE      =  4,
    RIF_STATIONTYPE_PASSENGERCAR    =  5,
    RIF_STATIONTYPE_BUS             =  6,
    RIF_STATIONTYPE_LIGHTTRUCK      =  7,
    RIF_STATIONTYPE_HEAVYTRUCK      =  8,
    RIF_STATIONTYPE_TRAILER         =  9,
    RIF_STATIONTYPE_SPECIALVEHICLES = 10,
    RIF_STATIONTYPE_TRAM            = 11,
    RIF_STATIONTYPE_ROADSIDEUNIT    = 15
};

/* VehicleRole */
/* ----------- */
enum Rif_VehicleRole {
    RIF_VEHICLEROLE_DEFAULT          =  0,   /* Default vehicle role as indicated by the
vehicle type. */
    RIF_VEHICLEROLE_PUBLICTRANSPORT  =  1,   /* Vehicle is used to operate public transport
service. */
    RIF_VEHICLEROLE_SPECIALTRANSPORT =  2,   /* Indication for special transport, e.g.
oversized trucks. */
    RIF_VEHICLEROLE_DANGEROUSGOODS   =  3,   /* Vehicle used for dangerous goods
transportation. */
    RIF_VEHICLEROLE_ROADWORK         =  4,   /* Vehicle used to realize roadwork or road
maintenance mission. */
    RIF_VEHICLEROLE_RESCUE           =  5,   /* Vehicle used for rescue purposes, e.g. as a
towing service. */
    RIF_VEHICLEROLE_EMERGENCY        =  6,   /* Vehicle used for emergency mission, e.g.
ambulance, fire brigade. */
    RIF_VEHICLEROLE_SAFETYCAR        =  7,   /* Vehicle is used for public safety, e.g.
patrol. */
    RIF_VEHICLEROLE_AGRICULTURE      =  8,   /* Vehicle is used for agriculture, e.g. farm
tractor. */
    RIF_VEHICLEROLE_COMMERCIAL       =  9,   /* Vehicle is used for transportation of
commercial goods. */
    RIF_VEHICLEROLE_MILITARY         = 10,   /* Vehicle is used for military purpose. */
    RIF_VEHICLEROLE_ROADOPERATOR     = 11,   /* Vehicle is used in road operator missions. */
    RIF_VEHICLEROLE_TAXI             = 12    /* Vehicle is used to provide an authorized taxi
service. */
};

/* VehicleSubRole */
/* -------------- */
enum Rif_VehicleSubRole {
    RIF_VEHICLESUBROLE_UNKNOWN        =  0,  /* Default vehicle role as indicated by the
vehicle type. */
    RIF_VEHICLESUBROLE_BUS            =  1,
    RIF_VEHICLESUBROLE_TRAM           =  2,
    RIF_VEHICLESUBROLE_METRO          =  3,
    RIF_VEHICLESUBROLE_TRAIN          =  4,
    RIF_VEHICLESUBROLE_EMERGENCY      =  5,  /* emergency vehicle with siren/lights. */
    RIF_VEHICLESUBROLE_SMOOTH         =  6,  /* ambulance smooth drive */
    RIF_VEHICLESUBROLE_TIMETABLE      =  7,  /* ambulance smooth drive */
    RIF_VEHICLESUBROLE_INTERVAL       =  8,  /* public transport time table service */
```

```
    RIF_VEHICLESUBROLE_EXPRESSTRANSIT =  9,
    RIF_VEHICLESUBROLE_NOSERVICE      = 10,  /* vehicles that are not in active service */
    RIF_VEHICLESUBROLE_PLATOON        = 11,  /* peleton, kolonne */
    RIF_VEHICLESUBROLE_ECODRIVING     = 12   /* EcoDriving vrachtwagens */

};


/* Direction - direction the vehicle is travelling in (ETSI: driveDirection) */
/* ------------------------------------------------------------------------- */
enum Rif_Direction {
    RIF_DIRECTION_FORWARD     =  0,
    RIF_DIRECTION_BACKWARD    =  1,
    RIF_DIRECTION_UNAVAILABLE =  2    /* Default direction */
};



/* CauseCode */
/* --------- */
enum Rif_CauseCode {
    RIF_CAUSECODE_UNKNOWN                                          =  0,
    RIF_CAUSECODE_TRAFFICCONDITION                                 =  1,
    RIF_CAUSECODE_ACCIDENT                                         =  2,
    RIF_CAUSECODE_ROADWORKS                                        =  3,
    RIF_CAUSECODE_ADVERSEWEATHERCONDITION_ADHESION                 =  6,
    RIF_CAUSECODE_HAZARDOUSLOCATION_SURFACECONDITION               =  9,
    RIF_CAUSECODE_HAZARDOUSLOCATION_OBSTACLEONTHEROAD              = 10,
    RIF_CAUSECODE_HAZARDOUSLOCATION_ANIMALONTHEROAD                = 11,
    RIF_CAUSECODE_HUMANPRESENCEONTHEROAD                           = 12,
    RIF_CAUSECODE_WRONGWAYDRIVING                                  = 14,
    RIF_CAUSECODE_RESCUEANDRECOVERYWORKINPROGRESS                  = 15,
    RIF_CAUSECODE_ADVERSEWEATHERCONDITION_EXTREMEWEATHERCONDITION  = 17,
    RIF_CAUSECODE_ADVERSEWEATHERCONDITION_VISIBILITY               = 18,
    RIF_CAUSECODE_ADVERSEWEATHERCONDITION_PRECIPITATION            = 19,
    RIF_CAUSECODE_SLOWVEHICLE                                      = 26,
    RIF_CAUSECODE_DANGEROUSENDOFQUEUE                              = 27,
    RIF_CAUSECODE_VEHICLEBREAKDOWN                                 = 91,
    RIF_CAUSECODE_POSTCRASH                                        = 92,
    RIF_CAUSECODE_HUMANPROBLEM                                     = 93,
    RIF_CAUSECODE_STATIONARYVEHICLE                                = 94,
    RIF_CAUSECODE_EMERGENCYVEHICLEAPPROACHING                      = 95,
    RIF_CAUSECODE_HAZARDOUSLOCATION_DANGEROUSCURVE                 = 96,
    RIF_CAUSECODE_COLLISIONRISK                                    = 97,
    RIF_CAUSECODE_SIGNALVIOLATION                                  = 98,
    RIF_CAUSECODE_DANGEROUSSITUATION                               = 99
};

/* DangerousGoods */
/* -------------- */
enum Rif_DangerousGoods {
    RIF_DANGEROUSGOODS_UNKNOWN                                          = -1,
    RIF_DANGEROUSGOODS_EXPLOSIVES1                                      =  0,
```

```
    RIF_DANGEROUSGOODS_EXPLOSIVES2                                          =  1,
    RIF_DANGEROUSGOODS_EXPLOSIVES3                                          =  2,
    RIF_DANGEROUSGOODS_EXPLOSIVES4                                          =  3,
    RIF_DANGEROUSGOODS_EXPLOSIVES5                                          =  4,
    RIF_DANGEROUSGOODS_EXPLOSIVES6                                          =  5,
    RIF_DANGEROUSGOODS_FLAMMABLEGASES                                       =  6,
    RIF_DANGEROUSGOODS_NONFLAMMABLEGASES                                    =  7,
    RIF_DANGEROUSGOODS_TOXICGASES                                           =  8,
    RIF_DANGEROUSGOODS_FLAMMABLELIQUIDS                                     =  9,
    RIF_DANGEROUSGOODS_FLAMMABLESOLIDS                                      = 10,
    RIF_DANGEROUSGOODS_SUBSTANCESLIABLETOSPONTANEOUSCOMBUSTION              = 11,
    RIF_DANGEROUSGOODS_SUBSTANCESEMITTINGFLAMMABLEGASESUPONCONTACTWITHWATER = 12,
    RIF_DANGEROUSGOODS_OXIDIZINGSUBSTANCES                                  = 13,
    RIF_DANGEROUSGOODS_ORGANICPEROXIDES                                     = 14,
    RIF_DANGEROUSGOODS_TOXICSUBSTANCES                                      = 15,
    RIF_DANGEROUSGOODS_INFECTIOUSSUBSTANCES                                 = 16,
    RIF_DANGEROUSGOODS_RADIOACTIVEMATERIAL                                  = 17,
    RIF_DANGEROUSGOODS_CORROSIVESUBSTANCES                                  = 18,
    RIF_DANGEROUSGOODS_MISCELLANEOUSDANGEROUSSUBSTANCES                     = 19
};


/* PriorityRequestType */
/* ------------------- */
enum Rif_PriorityRequestType {
    RIF_PRIORITYREQUESTTYPE_NONE         = 0,
    RIF_PRIORITYREQUESTTYPE_REQUEST      = 1,
    RIF_PRIORITYREQUESTTYPE_UPDATE       = 2,
    RIF_PRIORITYREQUESTTYPE_CANCELLATION = 3
};

/* PrioritizationState */
/* ------------------- */
enum Rif_PrioritizationState {
    RIF_PRIORITIZATIONSTATE_UNKNOWN           = 0,   /* Unknown state. */
    RIF_PRIORITIZATIONSTATE_REQUESTED         = 1,   /* This prioritization request was
detected by the traffic controller. */
    RIF_PRIORITIZATIONSTATE_PROCESSING        = 2,   /* Checking request (request is in
queue, other requests are prior). */
    RIF_PRIORITIZATIONSTATE_WATCHOTHERTRAFFIC = 3,   /* Cannot give full permission,
therefore watch for other traffic. Note that other requests may be present. */
    RIF_PRIORITIZATIONSTATE_GRANTED           = 4,   /* Intervention was successful and now
prioritization is active. */
    RIF_PRIORITIZATIONSTATE_REJECTED          = 5,   /* The prioritization request was
rejected by the traffic controller. */
    RIF_PRIORITIZATIONSTATE_MAXPRESENCE       = 6,   /* The request has exceeded maxPresence
time.
                                                        Used when the controller has
determined that the requester should then back off and request an alternative. */
    RIF_PRIORITIZATIONSTATE_RESERVICELOCKED   = 7    /* Prior conditions have resulted in a
reservice locked event:
                                                        the controller requires the
passage of time before another similar request will be accepted. */
};

/* TrafficDirection */
/* ---------------- */
```

```
enum Rif_TrafficDirection {
    RIF_TRAFFICDIRECTION_ALLTRAFFICDIRECTIONS = 0,
    RIF_TRAFFICDIRECTION_UPSTREAMTRAFFIC      = 1,
    RIF_TRAFFICDIRECTION_DOWNSTREAMTRAFFIC    = 2,
    RIF_TRAFFICDIRECTION_OPPOSITETRAFFIC      = 3
};

/* TurnIntention */
/* ------------- */
enum Rif_TurnIntention {
    RIF_TURNINTENTION_UNKNOWN  = 0,
    RIF_TURNINTENTION_LEFT     = 1,
    RIF_TURNINTENTION_STRAIGHT = 2,
    RIF_TURNINTENTION_RIGHT    = 3
};
```

```
/* TrustState */
/* ---------- */
enum Rif_TrustState {
    RIF_TRUSTSTATE_UNSECURED = 0,   /* no digital signature present     */
    RIF_TRUSTSTATE_UNTRUSTED = 1,   /* the digital signature is not trusted or cannot be
verified */
    RIF_TRUSTSTATE_TRUSTED   = 2    /* the digital signature is trusted */
};

/* -------------------------- */
/* ITS Station Interface (CAM) */
/* -------------------------- */
struct Rif_PublicTransport {
   rif_bool embarkation;
   rif_int  lineNr;
   rif_int  vehicleID;      /* Unique per company     */
   rif_int  serviceNr;      /* Same as block number   */
   rif_int  journeyNr;
   rif_int  supportNr;      /* Support journey number */
   rif_int  companyNr;
   rif_int  occupancy;      /* Number of passengers   */
};

struct Rif_SpecialTransport {
   rif_bool heavyLoad;
   rif_bool excessWidth;
   rif_bool excessLength;
   rif_bool excessHeight;
};

struct Rif_RoleAttributes {
   /* Deze velden zijn allemaal optioneel in de RIS-FI.
    * als ze niet bekend zijn, worden ze voor ints op -1 gezet en voor bools op false (0). */
   rif_bool                    lightBarActivated;
   rif_bool                    sirenActivated;
   enum Rif_CauseCode          incidentIndication;     /* CauseCode, zie definitie bovenaan.
*/
   rif_int                     incidentSubIndication;  /* SubCauseCode, groepeert suboorzaken
van verkeerssituaties. */
   /* Als de de public transport en special transport niet bekend zijn,
    * dan worden de velden binnen deze attributen voor ints op -1 gezet en voor bools op
false. */
   struct Rif_PublicTransport  publicTransport;
   struct Rif_SpecialTransport specialTransport;
   enum Rif_DangerousGoods     dangerousGoods;         /* DangerousGoods, zie definitie
bovenaan, -1 indien onbekend. */
};

struct Rif_PositionConfEllipse {
   rif_float semiMajorConfidence;   /* major axis of ellipse, -1.0 indien onbekend. */
   rif_float semiMinorConfidence;   /* minor axis of ellipse, -1.0 indien onbekend. */
   rif_float semiMajorOrientation;  /* orientation of the major axis relative to
                                     * navigation coordinate north,
                                     * -1.0 indien onbekend. */
```

```
};

/* This object represents the positionConfidenceEllipse received with CAM.
 * For horizontal positions a confidence area is used instead of a single confidence
 * interval.
 * The confidence area is specified via a major axis, minor axis and orientation of the
 * major axis relative to navigation coordinate north.
 * semiMajorConfidence, semiMinorConfidence: 1-40,93, where a value greater tan 40,93
 * indicates aout of range.
 * The value shall be set to null if the information is unavailable.
 * semiMajorOrientation: 0-360,0.The value shall be set to null if the information is
 * unavailable.
 * semiMajorConfidence, semiMinorConfidence: length in meters
 * semiMajorOrientation: degrees with regard to WGS84 north
 */

struct Rif_Location {      /* also used in ITS Events Interface (DENM) */
   rif_float latitude;     /* -90.0 t/m 90.0 degrees. */
   rif_float longitude;    /* -180.0 t/m 180.0 degrees. */
   rif_float elevation;    /* -100.0 t/m 8000.0 m, -101.0 indien onbekend. */
};



struct Rif_MapMatch {
   rif_string intersection;    /* Intersection ID, niet ingevuld ("") betekent geen map
match. */
   rif_int    lane;            /* Lane hoort bij een SG, als hier 0 is ingevuld betekent dat
deze zich op het conflict vlak (midden van het kruispunt) bevindt. */
   rif_string signalGroup;     /* SignalGroup ID is zonder Intersection ID.
                                * SignalGroup ID niet ingevuld (""), betekent onbekend. */
   /* T.o.v. de stopstreep van de SG, als deze waarde negatief is, dan is het een uitgaande
lane.
    * (-) 0 t/m 429496729.5 m, -9,999,999.0 indien onbekend. */
   rif_float  distance;
   rif_float  offset;       /* 0 t/m 429496729.5 m, -1.0 indien onbekend. */
};

struct Rif_ItsStation {
   rif_string                id;            /* ObjectID van het voertuig. */
   enum Rif_StationType      stationType;   /* StationType, zie definitie bovenaan. */
   rif_timestamp             locationTime;  /* Timestamp: het aantal milliseconden sinds 1-1-
1970 00:00:00:00 UTC.
                                             * Varieert van 0 tot 18446744073709509551615,
eenheid 1 milliseconde. */
   enum Rif_VehicleRole      role;          /* VehicleRole, zie definitie bovenaan. */
   rif_float                 length;        /* 0 t/m 429496729.5 m, -1.0 indien onbekend. */
   rif_float                 width;         /* 0 t/m 429496729.5 m, -1.0 indien onbekend. */
   enum Rif_Direction        direction;     /* (drive)Direction, zie definitie bovenaan.*/

   struct Rif_Location       location;
   /* Rijrichting/hoek/oriëntatie ten opzichte van WGS84 Noord, met de klok mee.
    * 0 t/m 360.0 graden, -1.0 indien onbekend. */
   rif_float                 heading;
   rif_float                 speed;         /* 0 t/m 99.0 m/s, -1.0 indien onbekend. */
```

```
   rif_float                acceleration;  /* -16.0 to 16.0 m/s^2, -17.0 indien onbekend. */
   struct Rif_RoleAttributes roleAttributes;
   struct Rif_PositionConfEllipse positionConfEllipse;
   enum Rif_TurnIntention    turn;          /* TurnIndication, zie definitie bovenaan. */
   /* Meer dan 1 lane beschikbaar.
    * Als de MapMatch niet ingevuld/valid is dan zal de intersection ID binnen een MapMatch
leeg zijn (""). */
   struct Rif_MapMatch       matches[RIF_MAXLANES];
   enum Rif_TrustState       trust;         /* TrustState, zie definitie bovenaan. */

   /* Geldigheid: 1 = valid, 0 = niet valid, verwijderen uit het buffer.
    * Behalve het id doen de andere velden er dan niet meer toe. */

   rif_string               signalGroup[RIF_MAXSIGNALGROUPSINCAM]; /* SignalGroup ID is
            * zonder Intersection ID. SignalGroup ID niet ingevuld (""), betekent
            * onbekend. The attribute signalGroup is the translation result from
            * the connections received with the corresponding SRM0 message. */

   rif_bool valid;
};




/* ------------------------------------ */
/* PrioritizationRequest Interface (SRM) */
/* ------------------------------------ */
struct Rif_TransitStatus {
   /* Optioneel, alle velden zijn "false" indien onbekend. */
   rif_bool loading;       /* parking and unable to move at this time */
   rif_bool anADAuse;      /* an ADA§ access is in progress, wheelchairs, kneeling, etc. */
   rif_bool aBikeLoad;     /* loading of a bicycle is in progress */
   rif_bool doorOpen;      /* a vehicle door is open for passenger access */
   rif_bool charging;      /* a vehicle is connected to charging point */
   rif_bool atStopLine;    /* a vehicle is at the stop line for the lane it is in */
};


struct Rif_PrioritizationRequest {
   rif_string                   id;             /* ID van de request zelf, deze bestaat uit
<itsStationID>_<requestID>. */
   rif_int                      sequenceNumber; /* Het vervolgnummer van deze request, deze
kan gebruikt worden als de request veranderd. */
   enum Rif_PriorityRequestType requestType;    /* Als er geen request is dan is hier de
waarde 0 ingevuld. */
   rif_string                   itsStation;     /* ObjectID van de bijbehorende itsStation
waarvoor deze request geldt. */
   rif_string                   intersection;   /* ObjectID van het kruispunt waarvoor deze
request geldt. */
   enum Rif_VehicleRole         role;           /* VehicleRole, zie definitie bovenaan. */
   enum Rif_VehicleSubRole      subrole;        /* VehicleSubRole, zie definitie bovenaan. */
   rif_timestamp                eta;            /* Timestamp: verwachte aankomsttijd van het
voertuig bij de stopstreep. */
                                                /* eta=0 – onbekend (unknown) */
   /* De richtingen */
```

```
   rif_string                  signalGroup;  /* SignalGroup ID is zonder Intersection ID.
                               * SignalGroup ID niet ingevuld (""), betekent onbekend. */
                              /* Indien beschikbaar, dan is dit een accurate richting.    */
   /* Nummer voor het groeperen van alle naderende rijstroken van een arm in één groep.
    * Fiets- en voetgangersovergangen die een kruispunt oversteken, hebben dezelfde approach
    * als het kruispunt die zij oversteken. 0 t/m 15, 0 indien onbekend. */
   rif_int                     approach;
   rif_string                  routeName;      /* Naam van de route, ("") indien onbekend.
*/
   struct Rif_TransitStatus    transitStatus;  /* Optioneel, alle velden zijn "false" indien
onbekend. */
   rif_int                     punctuality;    /* -3600 to 3600 s, negative values indicate
early arrival. -3601 als onbekend. */
   rif_int                     importance;     /* De belangrijkheid van het verzoek, -1
indien onbekend. */
   enum Rif_TrustState         trust;          /* TrustState, zie definitie bovenaan. */

   /* Geldigheid: 1 = valid, 0 = niet valid, verwijderen uit het buffer.
    * Behalve het id doen de andere velden er dan niet meer toe. */
   rif_bool valid;
};

/* ------------------------------------ */
/* ActivePrioritization Interface (SSM) */
/* ------------------------------------ */
struct Rif_ActivePrioritization {
   rif_string                  id;             /* ID van de priority request, om de active
prioritization hieraan te kunnen koppelen. */
   rif_int                     sequenceNumber; /* Zelfde als de sequencyNumber van de
bijbehorende priority request. */
   enum Rif_PrioritizationState prioState;     /* PrioritizationState, zie definitie
bovenaan. */
};



/* -------------------------- */
/* ITS Events Interface (DENM) */
/* -------------------------- */
struct Rif_Path {
   /* Deze struct beschrijft een pad d.m.v een verzameling punten. */
   /* Punten worden in de volgorde gedefinieerd vanaf de dichtstbijzijnde afstand van de
referentielocatie van het pad (bv. de stoplijn).*/
   /* De array wordt gelezen totdat een locatie met de waarde [0,0,0] wordt uitgelezen of
totdat MAXPOINTS wordt bereikt. */
   struct Rif_Location points[RIF_MAXPOINTS];
};

/* Deze struct beschrijft een geografisch gebied, afgebakend door een geometrische vorm. */
struct Rif_Area {
   struct Rif_Location centre;     /* Middelpunt van het gebied. */
   /* De majorAxis is de afstand tussen het middelpunt en de korte zijde van de geometrische
vorm (loodrecht op de korte zijde).
```

```
    * Voor een cirkel hebben de majorAxis en minorAxis dezelfde waarde (en circular heeft de
waarde "true"). */
   rif_float           majorAxis;  /* Length: 0 t/m 429496729.5 m, -1.0 indien onbekend. */
   rif_float           minorAxis;  /* Length: 0 t/m 429496729.5 m, -1.0 indien onbekend. */
   rif_float           angle;      /* Heading: de azimut hoek van de lange kant van de
geometrische vorm. 0 t/m 360,0, -1.0 indien onbekend. */
   rif_bool            circular;   /* Indien "false" dan is het gebied een rechthoekig gebied
in plaats van een cirkelvormig gebied. */
};


struct Rif_ItsEvent {
   /* ID van het ITS event, deze wordt ook gebruikt wanneer er wordt verwezen naar eerder
aangemaakte ITS events.
    * Bijvoorbeeld wanneer deze geupdate of verwijderd wordt. */
   rif_string              id;
   rif_timestamp           detectionTime;          /* Timestamp: het aantal milliseconden
sinds 1-1-1970 00:00:00:00 UTC.
                                                    * Varieert van 0 tot
18446744073709509551615, eenheid 1 milliseconde. */
   enum Rif_CauseCode      eventType;              /* Causecode, zie definitie bovenaan. */
   rif_int                 eventSubType;           /* Subcausecode */
   struct Rif_Location     eventPosition;          /* Lokatie van de ITS event. */
   rif_int                 validityDuration;       /* Duur van het event, 0 t/m 86400 s */
   rif_float               relevanceDistance;      /* 0 t/m 429496729.5 m, -1.0 indien
onbekend. */
   enum Rif_TrafficDirection trafficDirection;     /* TrafficDirection, zie definitie
bovenaan. */
   struct Rif_Path         traces[RIF_MAXTRACES];  /* Als er geen trace gebruikt wordt, dan
moet de eerste Location binnen een Path op [0, 0, 0] worden gezet. */
   /* Optioneel veld in de RIS-FI.
    * Indien onbekend of niet gebruikt dan moet de centre Location binnen de Area op [0, 0,
0] worden gezet. */
   struct Rif_Area         destinationArea;
   rif_int                 repetitionInterval;     /* 0 t/m 10000 ms, -1 indien onbekend.
*/
   enum Rif_TrustState     trust;                  /* TrustState, zie definitie bovenaan.
*/

   /* Geldigheid: 1 = valid, 0 = niet valid, verwijderen uit het buffer.
    * Behalve het id doen de andere velden er dan niet meer toe. */
   rif_bool valid;
   /* Interne referentie waarmee de applicatie zelf aangemaakte ITS events kan updaten of
verwijderen.
    * Door de applicatie zelf in te vullen. Voor ontvangen DENMs wordt -1 gebruikt. */
   rif_int  internalRef;
};
```

```
/* AdvisoryType */
/* ------------ */
enum Rif_AdvisoryType {
    RIF_ADVISORYTYPE_NONE       =  0,
    RIF_ADVISORYTYPE_GREENWAVE  =  1,
    RIF_ADVISORYTYPE_ECODRIVE   =  2,
    RIF_ADVISORYTYPE_TRANSIT    =  3
};

/* ------------ */
/* SpeedProfile */
/* ------------ */
struct Rif_SpeedProfile {
   enum Rif_AdvisoryType type;       /* AdvisoryType zie definitie hierboven.        */
   rif_float             distance;   /* 0 t/m 429496729.5 m, -1.0 indien onbekend.   */
   rif_float             speed;      /* 0 t/m 99.0 m/s, -1.0 indien onbekend.        */
};

/* ------------ */
/* QueueLength */
/* ------------ */
struct Rif_QueueLength {
   rif_int  zoneLength;         /* 0 t/m 10.000 m, 0 = no queue, -1 indien onbekend. */
                                /* 10.000=distances > 10.000 m                       */
   rif_int  lane;               /* laneID van de signalGroup bij de stopstreep.      */
                                /* -1 indien onbekend.                               */

/* ------------ */
/* Signalgroup */
/* ------------ */
struct Rif_SignalGroup {
   rif_string  id;           /* ObjectID - SignalGroup ID is zonder Intersection ID.*/
   struct Rif_SpeedProfile speedProfiles[RIF_MAXSPEEDPROFILES];  /* snelheidsadvies */
   struct Rif_QueueLength  queue[RIF_MAXQUEUES];                 /* wachtrijlengte  */
};

/* SignalHeadType - Traffic Lights including Abnormal Lights */
/* --------------------------------------------------------- */
/* normale verkeerslichten;    - hoofdlichten - waarden   1- 99 */
/* bijzondere verkeerslichten - hoofdlichten - waarden 101-199 */
/* bijzondere verkeerslichten - nevenlichten - waarden 201-299 */
/* bijzondere verkeerslichten - geen SPaT    - waarden 301-399 */

enum Rif_SignalHeadType {
   RIF_SIGNALHEADTYPE_UNKNOWN                      = 0,

/* Normale verkeerslichten - hoofdlichten */
/* -------------------------------------- */
   RIF_SIGNALHEADTYPE_TrafficLight                 =   1,
   RIF_SIGNALHEADTYPE_PedestrianLight              =   2,
   RIF_SIGNALHEADTYPE_PublicTransportLight         =   3,
   RIF_SIGNALHEADTYPE_SpecialPublicTransportLight  =   4,
   RIF_SIGNALHEADTYPE_OtherNormal                  =  99,
```

```
/*  Bijzondere verkeerslichten - hoofdlichten */
/* ---------------------------------------- */
   RIF_SIGNALHEADTYPE_WarningLightTramCrossing    = 101,
   RIF_SIGNALHEADTYPE_SpecialPedestrianLight      = 102,
   RIF_SIGNALHEADTYPE_TwoLensTrafficLight         = 103,
   RIF_SIGNALHEADTYPE_TwoLensTrafficLightSpecial  = 104,
   RIF_SIGNALHEADTYPE_TwoLensPedestrianLight      = 105,
   RIF_SIGNALHEADTYPE_MeteringLight               = 106,
   RIF_SIGNALHEADTYPE_OtherAbnormalMain           = 199,

/*  Bijzondere verkeerslichten - nevenlichten - bij signaalgroep fc[] */
/* ------------------------------------------------------------------ */
   RIF_SIGNALHEADTYPE_AmberFlashingLensRightTurn  = 201,
   RIF_SIGNALHEADTYPE_AmberFlashingLensLeftTurn   = 202,
   RIF_SIGNALHEADTYPE_GreenArrowLensRightTurn     = 203,
   RIF_SIGNALHEADTYPE_GreenArrowLensLeftTurn      = 204,
   RIF_SIGNALHEADTYPE_GreenArrowLensStraight      = 205,
   RIF_SIGNALHEADTYPE_RightTurnFreeForBicycles    = 206,
   RIF_SIGNALHEADTYPE_ClearanceArrowLeftTurn      = 207,
   RIF_SIGNALHEADTYPE_OtherAbnormalSide           = 299,

/*  Bijzondere verkeerslichten - geen SpaT */
/* -------------------------------------- */
   RIF_SIGNALHEADTYPE_GreenWaveSignal             = 301,
   RIF_SIGNALHEADTYPE_OtherNoSPaT                 = 399

};

/* --------------------------- */
/* SignalHead - Abnormal Lights */
/* --------------------------- */
struct Rif_SignalHead {
   enum Rif_SignalHeadType type;   /* SignalHeadType, zie definitie hierboven. */
   rif_int                 mainLightIndex;    /* signaalgroepindex hoofdlicht; */
                                              /* -1 indien niet gebruikt.      */
};

/* ----------------- */
/* ProductInformation */
/* ----------------- */
struct Rif_ProductInformation {
   rif_string  manufacturerName; /* The name of the manufacturer of a iTLC Component. */
   rif_string  certifiedName;    /* Certifed product name. */
   rif_string  certifiedVersion; /* Certifed product version. */
   rif_string  version;          /* Actual product version (relevant if the actual    */
                /* product version differs from the version on the certificate).      */
                /* Semantic versioning is mandatory for certifiedVersion and version. */
};


#endif /* _RIF_DEFS_H */
```

7. Appendix: rif.inc

```
/* RIS INTERFACE: PROCESBESTURING - APPLICATIEPROGRAMMA */
/* ================================================== */

/* RIS  :  versie 1.22.0    */
/* FILE :  rif.inc       */
/* DATUM:  12-06-2022 - versie 2.0 - Consolidatie uitbreidingen */
/* DATUM:  10-06-2020 - versie 1.2 - toelichting aangepast bij eta - in rif_defs.h */
/* DATUM:  05-10-2019 - versie 1.1 - RIF_UTC_TIME_PB toegevoegd */
/* DATUM:  12-09-2018 - versie 1.0    */


#ifndef _RIFDEF_INC
#define _RIFDEF_INC

/* include file */
/* ------------ */
  #include "rif_defs.h" /* declaratie RIS Interface objecten */

/* Definities grootte databuffers */
/* ------------------------------ */
   #define RIF_MAX_ITSSTATION_PB   100   /* Grootte buffer interface t.b.v. CAM berichten
procesbesturing    */
   #define RIF_MAX_PRIOREQUEST_PB   25   /*                          t.b.v. SRM berichten
procesbesturing    */
   #define RIF_MAX_ACTIVEPRIO_AP    10   /*                          t.b.v. SSM berichten
applicatie         */
   #define RIF_MAX_ITSEVENT_AP      10   /*                          t.b.v. DENM berichten
applicatie        */
   #define RIF_MAX_ITSEVENT_PB      10   /*                          t.b.v. DENM berichten
procesbesturing   */

#ifdef RIF_PUBLIC
   const rif_int RIF_ITSSTATION_PB_MAX  = RIF_MAX_ITSSTATION_PB;
   const rif_int RIF_PRIOREQUEST_PB_MAX = RIF_MAX_PRIOREQUEST_PB;
   const rif_int RIF_ACTIVEPRIO_AP_MAX  = RIF_MAX_ACTIVEPRIO_AP;
   const rif_int RIF_ITSEVENT_AP_MAX    = RIF_MAX_ITSEVENT_AP;
   const rif_int RIF_ITSEVENT_PB_MAX    = RIF_MAX_ITSEVENT_PB;
#else
   extern const rif_int RIF_ITSSTATION_PB_MAX;
   extern const rif_int RIF_PRIOREQUEST_PB_MAX;
   extern const rif_int RIF_ACTIVEPRIO_AP_MAX;
   extern const rif_int RIF_ITSEVENT_AP_MAX;
   extern const rif_int RIF_ITSEVENT_PB_MAX;
#endif

/* ITSSTATION_PB[]-buffer (CAM) */
/* ---------------------------- */
/* De buffer waar alle ITS stations in worden gezet. */
/* Deze wordt geschreven door de PB en gelezen door de AP. */
#ifdef RIF_PUBLIC
   struct Rif_ItsStation RIF_ITSSTATION_PB[RIF_MAX_ITSSTATION_PB];

   rif_int RIF_ITSSTATION_PB_WRITE;  /* Te schrijven door de PB */
```

```
    rif_int RIF_ITSSTATION_PB_READ;   /* Te schrijven door de AP */
#else
    extern struct Rif_ItsStation RIF_ITSSTATION_PB[];

    extern rif_int RIF_ITSSTATION_PB_WRITE;
    extern rif_int RIF_ITSSTATION_PB_READ;
#endif

/* PRIOREQUEST_PB[]-buffer (SRM) */
/* ----------------------------- */
/* De buffer waar alle prioritization requests in worden gezet. */
/* Deze wordt gelezen door de AP en geschreven door de PB. */
#ifdef RIF_PUBLIC
    struct Rif_PrioritizationRequest RIF_PRIOREQUEST_PB[RIF_MAX_PRIOREQUEST_PB];

    rif_int RIF_PRIOREQUEST_PB_WRITE; /* Te schrijven door de PB */
    rif_int RIF_PRIOREQUEST_PB_READ;  /* Te schrijven door de AP */
#else
    extern struct Rif_PrioritizationRequest RIF_PRIOREQUEST_PB[];

    extern rif_int RIF_PRIOREQUEST_PB_WRITE;
    extern rif_int RIF_PRIOREQUEST_PB_READ;
#endif

/* ACTIVEPRIO_AP[]-buffer (SSM) */
/* ---------------------------- */
/* De buffer waar alle active prioritizations in worden gezet. */
/* Deze wordt geschreven door de AP en gelezen door de PB. */
#ifdef RIF_PUBLIC
    struct Rif_ActivePrioritization RIF_ACTIVEPRIO_AP[RIF_MAX_ACTIVEPRIO_AP];

    rif_int RIF_ACTIVEPRIO_AP_READ;  /* Te schrijven door de PB */
    rif_int RIF_ACTIVEPRIO_AP_WRITE; /* Te schrijven door de AP */
#else
    extern struct Rif_ActivePrioritization RIF_ACTIVEPRIO_AP[];

    extern rif_int RIF_ACTIVEPRIO_AP_READ;
    extern rif_int RIF_ACTIVEPRIO_AP_WRITE;
#endif

/* ITSEVENT_AP[]-buffer (DENM) */
/* --------------------------- */
/* De buffer waar alle ITS events vanuit de AP in worden gezet. */
/* Deze wordt geschreven door de AP en gelezen door de PB. */
#ifdef RIF_PUBLIC
    struct Rif_ItsEvent RIF_ITSEVENT_AP[RIF_MAX_ITSEVENT_AP];

    rif_int RIF_ITSEVENT_AP_READ;  /* Te schrijven door de PB */
    rif_int RIF_ITSEVENT_AP_WRITE; /* Te schrijven door de AP */
#else
    extern struct Rif_ItsEvent RIF_ITSEVENT_AP[];

    extern rif_int RIF_ITSEVENT_AP_READ;
    extern rif_int RIF_ITSEVENT_AP_WRITE;
#endif
```

```
/* ITSEVENT_PB[]-buffer (DENM) */
/* --------------------------- */
/* De buffer waar alle ITS events vanuit de PB in worden gezet. */
/* Deze wordt geschreven door de PB en gelezen door de AP. */
/* De events die hier in komen zijn alleen de events van andere applicaties. */
#ifdef RIF_PUBLIC
   struct Rif_ItsEvent RIF_ITSEVENT_PB[RIF_MAX_ITSEVENT_PB];

   rif_int RIF_ITSEVENT_PB_WRITE; /* Te schrijven door de PB */
   rif_int RIF_ITSEVENT_PB_READ;  /* Te schrijven door de AP */
#else
   extern struct Rif_ItsEvent RIF_ITSEVENT_PB[];


   extern rif_int RIF_ITSEVENT_PB_WRITE;
   extern rif_int RIF_ITSEVENT_PB_READ;
#endif


/* RIF_UTC_TIME_PB */
/* --------------- */
/* Timestamp RIF_UTC_TIME_PB is de actuele tijd. */
/* Het aantal milliseconden sinds 1-1-1970 00:00:00:00 UTC. */
/* Deze UTC-tijd wordt geschreven door de PB en gelezen door de AP. */

#ifdef RIF_PUBLIC
   rif_timestamp RIF_UTC_TIME_PB;   /* Te schrijven door de PB */
#else
   extern rif_timestamp RIF_UTC_TIME_PB;
#endif


/* RIF_SIGNALGROUP_AP[]-buffer */
/* --------------------------- */
/* RIF_SIGNALGROUP_AP[] bevat de snelheidsprofielen (type, distance, speed) en     */
/* wachtrijlengten (zoneLength, lane) van de signaalgroepen van de ITS-applicatie. */
/* Met de wijzigingsvlag RIF_SIGNALGROUP_AP_SPEEDPROFILE_CHNG[] wordt aangegeven   */
/* dat het snelheidsprofiel van een signaalgroep is gewijzigd.                     */
/* Met de wijzigingsvlag RIF_SIGNALGROUP_AP_QUEUELENGTH_CHNG[] wordt aangegeven    */
/* dat de wachtrijlengte van een signaalgroep is gewijzigd.                        */

#ifdef RIF_PUBLIC
   struct Rif_SignalGroup RIF_SIGNALGROUP_AP[FCMAX];  /* FCMAX = CIF_AANT_US_FC */
#else
   extern struct Rif_SignalGroup RIF_SIGNALGROUP_AP[];
#endif

#ifdef RIF_PUBLIC
   rif_int RIF_SIGNALGROUP_AP_SPEEDPROFILE_CHNG[FCMAX]; /* SpeedProfile gewijzigd */
   rif_int RIF_SIGNALGROUP_AP_QUEUELENGTH_CHNG[FCMAX];  /* QueueLength gewijzigd  */
```

```
  #else
    extern rif_int RIF_SIGNALGROUP_AP_SPEEDPROFILE_CHNG[];
    extern rif_int RIF_SIGNALGROUP_AP_QUEUELENGTH_CHNG[];
#endif

/* RIF_SIGNALHEAD_AP[]-buffer */
/* -------------------------- */
/* RIF_SIGNALHEAD_AP[] bevat het type verkeerslantaarn (type, mainLightIndex) van   */
/* de signaalgroepen van de ITS-applicatie. het type verkeerslantaarn is van belang */
/* voor de bijzondere lichten (Abnormal Lights).                                    */
/* RIF_SIGNALHEAD_AP[] wordt geschreven door de AP en gelezen door de PB.           */

#ifdef RIF_PUBLIC
    struct Rif_SignalHead RIF_SIGNALHEAD_AP[FCMAX];     /* FCMAX = CIF_AANT_US_FC */
#else
    extern struct Rif_SignalHead RIF_SIGNALHEAD_AP[];
#endif


/* RIF_ITSINFO_AP */
/* -------------- */
/* RIF_ITSINFO_AP bevat de productinformatie (manufacturerName, certifiedName, */
/* certifiedVersion, version) van de ITS-Applicatie.                            */
/* RIS_ITSINFO_AP wordt geschreven door de AP en gelezen door de PB.            */

#ifdef RIF_PUBLIC

/* const struct Rif_ProductInformation RIF_ITSINFO_AP; //definitie in applicatie */


/* voorbeeld
 * ---------
 * const struct Rif_ProductInformation RIF_ITSINFO_AP = {
 *          "gemeente Rotterdam",      // manufacturerName,
 *          "TLCGen",                  // certifiedName
 *          "9.5.0",                   // certifiedVersion,
 *          "9.5.0"                    // version
 * };
 */

#else
    extern const struct Rif_ProductInformation RIF_ITSINFO_AP;
#endif


#endif /* _RIFDEF_INC */
```

8. Appendix: rif_facade.h

```
#ifndef _RIF_FACADE_H
#define _RIF_FACADE_H

#include <stddef.h>
#include "rif_defs.h"

/* -------------------- */
/* RIS Interface Facade */
/* -------------------- */
/* Het '_Type' moet gelijk zijn aan een van de RIS-objecten. */

template<class _Type>
class Rif_Callback {
public:
    /* Definitie van de notificatie-functie. */
    /* Deze functie zal door de PB worden aangeroepen voor elk ontvangen RIS-object. */
    virtual void OnNotifyObject(const _Type& object) = 0;
};

template<class _Type>
class Rif_Facade {
public:
    /* Voor elk _Type is er één facade beschikbaar. */
    static Rif_Facade<_Type>& get() {
        static Rif_Facade<_Type> instance;
        return instance;
    }
```

```
    /* Methode voor het aanmelden voor notificaties van ontvangen RIS-objecten. */
    void subscribe(Rif_Callback<_Type>* callback);
    /* Methode voor het verzenden van RIS-objecten. */
    void publish(const _Type& object);

private:
    Rif_Facade() {}                    // Constructor is niet publiekelijk!
    Rif_Facade(Rif_Facade const&);     // Kopiëren is niet toegestaan!
    void operator=(Rif_Facade const&); // Toewijzen is niet toegestaan!
};
```

```
template<class _Type>
class FunctionCallback : public Rif_Callback<_Type> {
public:
    /* Definitie van de functie-pointer. */
    typedef void (*Callback)(_Type const&);

    /* Wrap een functie-pointer. */
    static Rif_Callback<_Type>* Wrap(Callback callback) {
        if (callback != NULL) {
            return new FunctionCallback<_Type>(callback);
        }
        return NULL;
    }
private:
    Callback callback_; // Referentie naar de callback-functie.

    /* Construeer een Rif_Callback object van een functie-pointer. */
    FunctionCallback(Callback callback) : callback_(callback) {}

    /* Geef het ontvangen object door aan de callback-functie. */
    void OnNotifyObject(const _Type& object) {
        callback_(object);
    }
};

#endif /* _RIF_FACADE_H */
```