Koppelvlak Open DRIS

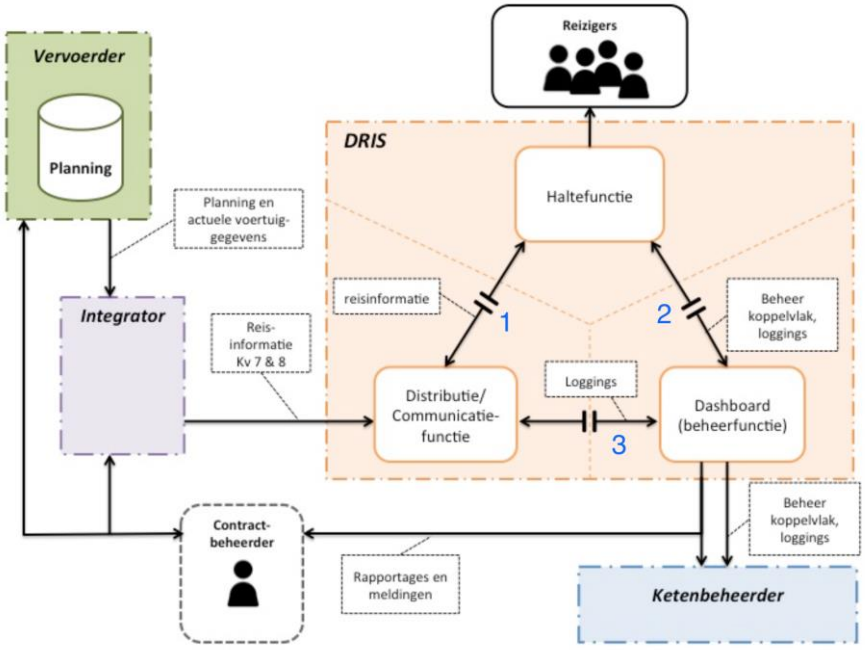
Versie 3.4 – 26 februari 2021

Wijzigingshistorie

Versie	Datum	Reden
3.1	25-06-21	Gebruikt als basis voor aanbesteding OV-Oost
3.2	06-08-20	Interne overdracht
3.3	23-10-20	Gebruikt bij aanbesteding Utrecht
3.4	26-02-21	Diverse tekstuele wijzigingen n.a.v. project OV-Oost

Inleiding

De architectuur van Open DRIS voorziet in een driehoek tussen beheer-, distributie- en haltesystemen. Hierin zijn drie koppelvlakken nodig, namelijk die tussen het distributiesysteem en de haltesystemen (1 in onderstaand figuur), tussen de dashboard- en de haltesystemen (2) en tenslotte tussen het distributiesysteem en de dashboardsystemen (3). Dit document beschrijft deze drie koppelvlakken.



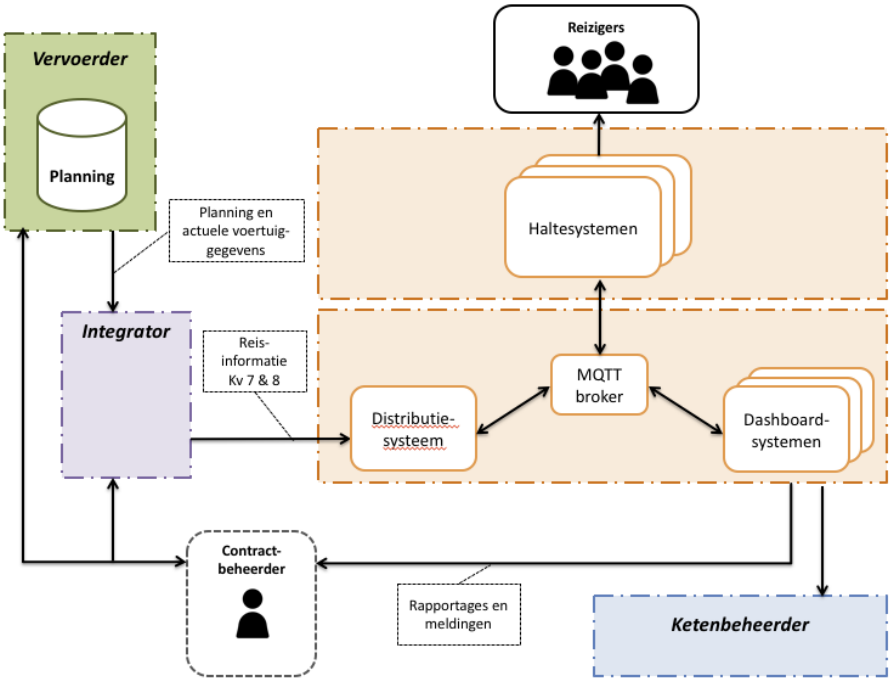
Reisinformatieketen voor DRIS

Het koppelvlak tussen het distributiesysteem en de haltesystemen voorziet voornamelijk in het doorgeven van reisinformatie vanuit de Dataleverancier en het aanmelden op de juiste subset die van belang is voor een haltesysteem. Omdat er veel verschillende verschijningsvormen en informatiebehoeftes zijn, speelt filteren en specificeren welke velden van belang zijn voor het haltesysteem een grote rol.

Het koppelvlak tussen het distributiesysteem en de dashboardsystemen is relatief eenvoudig en is bedoeld voor het communiceren van noodzakelijke logging en statusinformatie.

Het koppelvlak tussen halte- en dashboardsystemen bevat status- en diagnose-informatie van de haltesystemen. Met behulp van deze informatie, kunnen allerlei partijen een eigen dashboard maken. Het is uitdrukkelijk de bedoeling dat er ook dashboardsystemen komen die alle aangesloten haltesystemen beheren.

De drie soorten systemen communiceren met elkaar via het MQTT 5 protocol en kunnen op die manier real-time push-driven informatie met elkaar uitwisselen. Alle communicatie tussen de actoren in deze architectuur verloopt via een centrale MQTT broker.



Daarnaast moeten alle berichten gedefinieerd en geserialiseerd worden in het Protobuf 3 formaat. Hierdoor zijn berichten compact en op veel verschillende platformen toepasbaar door beschikbare standaard implementaties.

De rest van dit document behandelt de verschillende functionele flows in de koppelvlakken, de inhoud van de verschillende berichten die nodig zijn voor deze flows en tenslotte worden de MQTT details behandeld die nodig zijn voor het versturen van berichten.

Flows

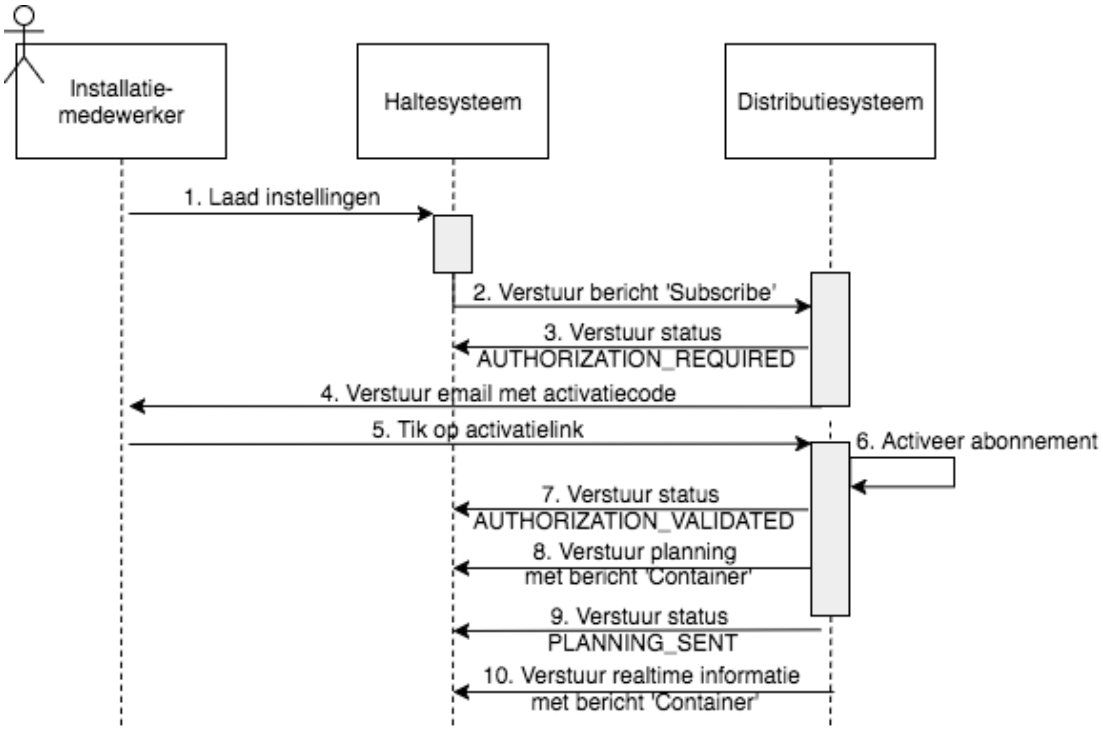
Aan- en afmelden

Het initiatief om een haltesysteem (weer) op te nemen in het systeem ligt bij het haltesysteem. Door het sturen van een aanmeldbericht abonneert het haltesysteem zich bij het distributiesysteem op de informatie van een of meer haltes, aangeduid met Quay-codes of StopPlace-codes zoals beschreven in het centraal halte bestand (CHB). Met hetzelfde aanmeldbericht maakt het haltesysteem zich ook bekend bij de dashboardsystemen. Die systemen gebruiken de Quay- of StopPlace-codes in het aanmeldbericht om praktische informatie zoals haltenaam en straatnaam op te halen uit het CHB.
Aanmelden gebeurt op basis van een client id (zie hoofdstuk MQTT voor details). Verder bevat het aanmeldbericht een aantal optionele velden, met deze velden wordt het distributiesysteem geïnformeerd over welke gegevens het haltesysteem wil ontvangen. Per veld kan gekozen worden voor: geen gegevens of alle wijzigingen.

Indien een haltesysteem zich voor de eerste keer aanmeldt, is het haltesysteem, naast het client id in de topicnaam, verplicht zich te identificeren met een emailadres van de beheer- of installatiepartij. Dit emailadres hoeft niet uniek te zijn. Ter autorisatie wordt door het distributiesysteem eenmalig per emailadres plus client id combinatie, een email gestuurd naar het emailadres van de leverancier van het haltesysteem ter verificatie. Dit emailbericht bevat een link waarmee het haltesysteem geautoriseerd wordt om data te ontvangen van het distributiesysteem. Na de verificatie wordt het haltesysteem direct actief. In

het aanmeldbericht kan ook een omschrijving worden ingevuld zodat verschillende systemen uit elkaar kunnen worden gehouden.

Op het aanmeldbericht wordt in het distributiesysteem een aantal controles uitgevoerd: alle verplichte velden moeten aanwezig zijn, Quay- of StopPlace-codes moeten bestaan in het CHB en het emailadres moet geldig zijn (per leverancier van Haltesystemen wordt een autorisatie-adres afgesproken en vastgelegd in het Distributiesysteem). Preconditie voor het aanmelden van een haltesysteem is dus dat de haltes waarover het haltesysteem informeert Quay-codes hebben in het CHB. Op alle aanmeldverzoeken wordt een antwoord teruggestuurd naar het haltesysteem met de status van de aanmelding (bericht 'SubscriptionResponse'). In geval van een foutief emailadres of ontbrekende informatie in het aanmeldbericht wordt de status 'REQUEST_INVALID' gecommuniceerd. Bij een stopcode die niet in CHB bekend is wordt de status 'STOP_INVALID' gecommuniceerd.



Het aanmeldproces: eerste keer per haltesysteem

Na het doorlopen van de autorisatieprocedure stuurt het distributiesysteem de status 'AUTHORISATION_VALIDATED' om aan te geven dat de autorisatie gelukt is. Daarna wordt de meest actuele versie van de in het distributiesysteem bekende ritten opgehaald en eventueel geldige en toekomstige vrije teksten opgehaald en naar het haltesysteem verstuurd. Indien er ritten bekend zijn voor het betreffende haltesysteem, wordt na het versturen van de ritten de status 'PLANNING_SENT' gestuurd. Indien er geen geplande ritten zijn in de komende periode wordt de status 'NO_PLANNING' gecommuniceerd. In dit geval is het abonnement echter wel actief en kunnen er op elk moment, ritten dan wel vrije teksten naar het haltesysteem worden gestuurd.

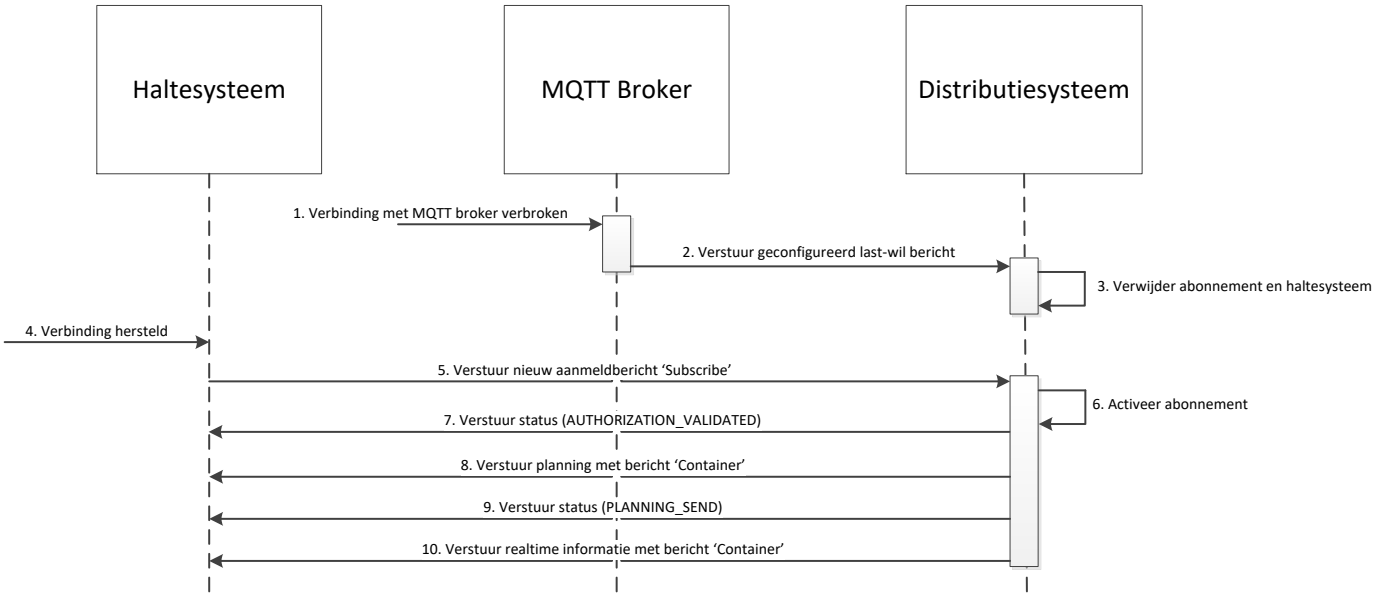
Dashboardsystemen kunnen zich ook abonneren op de aanmeldberichten en de antwoorden, zodat ze het aanmeldproces kunnen volgen en vastleggen.

Indien een haltesysteem zich twee keer aanmeldt zonder zich tussendoor af te melden, wordt dit door het distributiesysteem afgehandeld als een afmeldactie, gevolgd door opnieuw aanmelden. Hierbij zal opnieuw de geplande en actuele ritinformatie alsmede de actieve en geplande vrije teksten worden verstuurd.

Afmelden

Bij het opzetten van de verbinding met de broker configureert het haltesysteem het bericht Unsubscribe als een zogenaamd ‘last-will’ bericht. Indien de verbinding tussen haltesysteem en de MQTT broker wegvalt, publiceert de MQTT broker dit last-will bericht, dat door het distributiesysteem en de dashboardsystemen wordt ontvangen. Indien het distributiesysteem een last-will/afmeldbericht ontvangt zal het geen informatie meer sturen naar het haltesysteem tot het opnieuw is aangemeld. De dashboardsystemen zullen de status van het haltesysteem veranderen naar offline.

Voor het last-will bericht wordt het bericht Unsubscribe gevuld met ‘is_permanent’ = false en De timestamp wordt gevuld met het huidige tijdstip.



Het afmeldproces en gevolgd door het opnieuw aanmelden van het al eerder geautoriseerde haltesysteem

Om eventuele race-condities te voorkomen is het van belang dat het haltesysteem zich ook abonneert op zijn eigen unsubscribe berichten (bijvoorbeeld verstuurd door de broker als last will bericht), en unsubscribe berichten van het distributiesysteem. Na het ontvangen van een eigen unsubscribe bericht of een unsubscribe bericht van het distributiesysteem, moet het haltesysteem een nieuw aanmeldbericht sturen.

Het is daarnaast ook mogelijk om een haltesysteem permanent af te melden door middel van het Unsubscribe bericht, met waarde ‘is_permanent’ = true en het (huidige) tijdstip van afmelden. Dit is van belang indien een haltesysteem wordt verwijderd, gedeactiveerd of wordt vervangen. Dit voorkomt bijvoorbeeld extra logging van gegevens over een haltesysteem dat defect is en maakt (keten)beheer makkelijker. Indien een haltesysteem is afgemeld met ‘is_permanent’ = true en het haltesysteem meldt zich later weer aan (met dezelfde client id), moet het opnieuw de autorisatie via email doorlopen.

Reisinformatie

Op basis van het selectiefilter wordt er een lijst van passeertijden in unix-timestamps verstuurd van distributie- naar haltesysteem. Hierdoor is de afnemer niet afhankelijk van operationele dagen, slechts het begrip een tijdstip in UTC om naar een tijdzone van het haltesysteem te converteren is van belang, in dit geval: Europe/Amsterdam. Deze passeertijden worden geïdentificeerd door het veld ‘pass_time_hash’, dat

constant blijft gedurende een rit. Voor de unix-timestamp wordt de variant gebruikt waarvan de waarde 0 gelijkstaat aan de datum 1-1-1970 00:00:00 GMT.

Het distributiesysteem stuurt voor een te bepalen periode (parameter), ritten naar het haltesysteem, zodat het haltesysteem in geval van uitval van het systeem of verbinding toch reisinformatie kan tonen op basis van de dienstregeling. De exacte implementatie hiervan is opgenomen in het distributiesysteem en is te configureren.

Na het actief worden van een haltesysteem door middel van een succesvolle aanmelding, ontvangt het haltesysteem voor de komende periode geplande tijden, eventueel geactualiseerd met de tot dan toe bekende informatie. Daarnaast worden realtime updates meteen doorgestuurd. Beide typen berichten zijn voor een haltesysteem niet te onderscheiden en worden gekenmerkt door een zelfde hashsleutel (pass_time_hash).

Er zijn gevallen mogelijk waarbij er binnen de geldige geplande periode geen geplande ritten zullen plaatsvinden (bijvoorbeeld in het weekend, evenementenhaltes, etc.) en er ook geen geldige vrije teksten actief zijn op dat moment. Hierbij kan een geval ontstaan dat er geen reisinformatie voor het haltesysteem beschikbaar is. In dit geval wordt een statusbericht verstuurd naar het haltesysteem met status ‘NO_PLANNING’

Standaard berichtafhandeling
Ieder bericht wordt gecodeerd via protocol buffers. Het uitwisselingsformaat zorgt ervoor dat datatypes horizontaal worden *gepacked*. Er zijn daardoor minder bits nodig als het datatype niet volledig wordt gebruikt. Daarnaast kiezen we binnen het protocol voor het principe van uitwisselen in kolommen. De meeste databases maken gebruik van rijen, 1 rij komt vaak overeen met de attributen van 1 object. Door niet in rijen, maar in kolommen uit te wisselen krijgen we een tweede optimalisatie: informatie die zich herhaalt (denk aan: bestemmingen en publieke lijnnummers) staan dicht bij elkaar en kunnen slimmer worden gecomprimeerd. In een kolom georiënteerde database hebben alle kolommen hetzelfde aantal elementen. Een rij is op basis van de index samen te voegen. Element 1 uit kolom 1, komt daarmee overeen met element 1 uit kolom 2. Het haltesysteem heeft bij het aanmelden geselecteerd welke kolommen het wil ontvangen, alleen deze kolommen worden door het distributiesysteem gesynchroniseerd.

Het synchronisatieproces gaat uit van van een vaste identificatie voor een rit, dit in de vorm van een hashsleutel (‘pass_time_hash’). Op deze manier is iedere passeertijd uniek identificeerbaar en overschrijfbaar. Op basis van het selectiefilter worden de actuele passeertijden naar het haltesysteem verstuurd.

```
passing_times {
  pass_time_hash: 1487911693
  pass_time_hash: 3375514872
  target_departure_time: 1496090100
  target_departure_time: 1496086440
  expected_departure_time: 1496090100
  expected_departure_time: 1496086457
  trip_stop_status: PLANNED
  trip_stop_status: DRIVING
  destination {
    destination_name: "Marconiplein"
    destination_detail: “”
```

```
  }
  destination {
    destination_name: "Marconiplein"
    destination_detail: ‘’
  }
  line_public_number: "23"
  line_public_number: "23"
  generated_timestamp: 1496086082
  generated_timestamp: 1496086082
}
general_messages {}
```

Voorbeeld bericht ‘Container’ gevuld met actuele reisinformatie voor twee ritten

Vrije teksten

Naast passeertijden kunnen ook vrije teksten gecommuniceerd worden naar een haltesysteem. De berichten waarmee vrije teksten worden verstuurd hebben ook een hash en zijn uniek te identificeren met het veld ‘message_hash’. De vrije tekst kan worden verwijderd met het ‘GeneralMessageRemove’ bericht.Het haltesysteem is verantwoordelijk voor het tonen van de vrije tekst tussen de message_start_time en de message_end_time, of tot de tekst wordt gewist. Elke vrije tekst wordt los verstuurd naar het haltesysteem, indien er meerdere vrije teksten geldig zijn, bepaalt het haltesysteem hoe dit getoond wordt.

Overzichtdisplay

Indien een haltesysteem een overzicht moet tonen van meerdere quays zijn hier twee mogelijkheden voor.

a) Het haltesysteem meldt zich aan op meerdere quays: alle QuayCodes (NL:Q:xxxxxxxxx) die onderdeel zijn van de StopPlace waarvoor het overzichtdisplay geldt, worden opgenomen in het aanmeldbericht. Reisinformatie en vrije teksten voor alle gevraagde quay codes worden naar het haltesysteem verstuurd. Voordeel van deze aanpak is dat bijvoorbeeld een uitstaphalte die onderdeel is van een StopPlace kan worden uitgesloten.
b) Het haltesysteem meldt zich aan op een StopPlace: één (of eventueel meerdere) StopPlace(s) worden opgenomen in het aanmeldbericht (NL:S:xxxxxxxxx). Op basis van CHB wordt voor de geldige Quay Codes reisinformatie verstuurd. Dit heeft als voordeel dat indien de samenstelling van de StopPlace wijzigt en het Centrale Halte Bestand wordt aangepast, het distributiesysteem dit opnieuw inleest en het overzichtdisplay automatisch meegaat.

Client Id

Als bijlage van dit document is het bestand DrisKoppelVlak(versie).proto toegevoegd. Dit bestand is leidend voor het koppelvlak.
In dit hoofdstuk worden de verschillende ‘messages’ verder uitgewerkt.

ClientId:

	Veld	**Aantal**	**Uitleg**
ClientId	subscriber_owner_code	(1)	De SubscriberOwnerCode wordt gevuld met een waarde uit een overkoeppelende lijst. In deze lijst worden alle partijen beschreven die de distributie-, dashboard- of haltesystemen leveren. Zie tabel in bijlage 1. Deze bijlage wordt door DOVA OV_Data beheerd.
	subscriber_type	(1)	Geeft het type van de subscriber aan 0=distributiesysteem, 1=dashboardsysteem en 2=haltesysteem.
	serial_number	(1)	Volgnumer dat de combinatie van de velden uniek maakt. Voor een haltesysteem wordt geadviseerd om het serienummer van de hardware te gebruiken.

Berichten Koppelvlak Distributiesysteem - Haltesysteem

Subscribe

		Veld	Aantal	Uitleg
Subscribe	client_id		(1)	Id waarmee client zich aanmeld.
	stop_code		(1..n)	Een of meer quay codes uit het Centrale Halte Bestand (CHB) waarop het haltesysteem zich wil abonneren en reisinformatie en vrije teksten van wil ontvangen, beginnend met “NL:Q” of “NL:S”. (verplicht)
	display_properties		(0..1)	Geeft de kenmerken van het ontvangende haltesysteem aan
	DisplayProperties	text_characters	(0..1)	Het aantal tekens dat het display van het haltesysteem beschikbaar heeft voor de bestemmingsnaam. Het distributiesysteem kan daarmee bepalen welke bestemmingstekst (lengte) en detailbestemmingstekst gestuurd moeten worden. Indien deze waarde niet wordt gespecificeerd, wordt de langste waarde meegestuurd. Dit is van toepassing op haltesystemen die geen (praktische) lengtebeperking hebben.
		overview_display	(1)	Geeft aan of het display van dit Haltesysteem fungeert als overzichtdisplay (true) or not (false).
		destination_determination	(1)	Geeft aan welke bestemmingsteksten naar het haltesysteem moeten worden verstuurd: - MAX_CHARACTERS het haltesysteem wil een bestemmingstekst en detail bestemmingtekst ontvangen met maximaal de lengte als gegeven in tekst_characters. Het distributiesysteem stuurt de eerstpassende bestemmingstekst en detailbestemmingstekst. (als tekst_characters = 18 wordt een bestemmingstekst 16 gestuurd.) - SELF_DETERMING het haltesysteem wil alle bestemmingsteksten en detailbestemmingsteksten ontvangen die het distributiesysteem heeft en selecteert zelf welke tekst wordt getoond, rekening houdend met de mogelijkheden van de schermen.
	filter_parameters		(0..1)	Geeft de parameters (T1, T2, P1, P2) waarmee de KV8 reisinformatie gefilterd moet worden

				door het distributiesysteem.
	FilterParameters	filter_on	(0..1)	Geeft aan of er wel of niet gefilterd moet worden.
		waitingtime_low	(0..1)	De ondergrens voor de wachttijd (T1).
		waitingtime_high	(0..1)	De bovengrens voor de wachttijd (T1).
		percentage_low	(0..1)	Het percentage (P1) waarmee de wachttijd af moet wijken onder de ondergrens van de wachttijd (T1).
		percentage_high	(0..1)	Het percentage (P2) waarmee de wachttijd af moet wijken boven de ondergrens (T1).
	filter_filter		(0..1)	Geeft aan welke gegevens van een rit gestuurd moeten worden naar het haltesysteem. Elk veld van dit object correspondeert met een gelijknamige kolom in het bericht 'PassingTimes' waarvan wordt aangegeven hoe dit moet worden verstuurd. Geldige waardes zijn opgenomen in de enumeratie 'Delivery'. Mogelijk zijn: - NEVER: standaardwaarde. Stuur dit veld nooit. (Protobuff 3 kent geen optionele velden. Als een zender (repeated) velden leeg laat, wordt er geen data voor deze velden in het bericht geplaatst. Echter bij het deserialiseren aan de ontvangende kant worden de velden gevuld met een defaultwaarde, zoals gespecificeerd in de protobuff documentatie.) - ALWAYS: in elk bericht wordt de meest recente waarde gestuurd. Elke onderstaand veld spreekt voor zich: de waarde van het veld is in het PassingTimes bericht gedocumenteerd.
	FieldFilter	target_arrival_time	(0..1)	De aankomsttijd volgens dienstregeling. Deze kan meegestuurd worden, bijvoorbeeld om vertraging relatief aan die tijd te tonen
		target_departure_time	(0..1)	De vertrektijd volgens dienstregeling. Deze kan meegestuurd worden, bijvoorbeeld om vertraging relatief aan die tijd te tonen
		expected_arrival_time	(0..1)	De verwachte aankomsttijd. Kan gebruikt worden voor een haltesysteem dat aankomsttijden toont.
		expexted_departure_time	(0..1)	De verwachte vertrektijd. Kan gebruikt worden voor een haltesysteem dat vertrektijden toont.
		number_of_coaches	(0..1)	Dit geeft aan hoeveel (gekoppelde)

				rijtuigen/voertuigen op een rit worden ingezet. Dit kan getoond bij metro/tram haltes, maar kan ook bij andere modaliteiten.
		trip_stop_status	(0..1)	Dit veld geeft aan wat de status van de rit is. Displays die alleen vertrektijden volgens dienstregeling weergeven, zouden ervoor kunnen kiezen dit veld niet te ontvangen.
		transport_type	(0..1)	Onder andere nodig om het juiste icoon te tonen als het voertuig op de halte aankomt of de wachttijd 0 minuten is. E.e.a. conform de Mijksenaar richtlijn.
		wheelchair_accessible	(0..1)	Hiermee kan op een display getoond worden of het voertuig van een rit toegankelijk is voor rolstoelen.
		is_timingstop	(0..1)	Hiermee bepaalt het haltesysteem welke informatie getoond wordt na aankomst van het voertuig op de halte.
		stop_code	(0..1)	Relevant voor haltesystemen geabonneerd op meerdere quays, waarbij het van belang is te weten op welke quay een vertrek betrekking heeft.
		destinations	(0..1)	
		show_cancelled_trip	(0..1)	
		block_code	(0..1)	
		occupancy	(0..1)	
		line_public_number	(0..1)	
		side_code	(0..1)	
		line_direction	(0..1)	Relevant om te kunnen filteren op het aantal te tonen ritten per lijn en richting.
		line_color	(0..1)	Relevant voor haltesystemen die een lijnachtergrondkleur kunnen tonen
		line_text_color	(0..1)	Relevant voor haltesystemen die een lijnvoorgrondkleur kunnen tonen
		line_icon	(0..1)	Relevant voor haltesystemen die een icoon of afbeelding kunnen tonen
		destination_color	(0..1)	Relevant voor haltesystemen die een bestemmingsachtergrondkleur kunnen tonen
		destination_text_color	(0..1)	Relevant voor haltesystemen die een bestemmingsvoorgrondkleur kunnen tonen

		destination_icon	(0..1)	Relevant voor haltesystemen die een bestemmingsicoon of afbeelding kunnen tonen
		generated_timestamp	(0..1)	Relevant om eventueel “out of order” berichten te kunnen detecteren en te kunnen verifiëren dat altijd de laatste informatie wordt gebruikt
		journey_number	(0..1)	Ritnummer voor eventueel traceren van een rit op verschillende haltes
	description		(0..1)	Een omschrijving van de locatie van het haltesysteem of een begeleidende tekst. Wordt door het distributiesysteem meegestuurd in het autorisatieproces en kan door het dashboard worden getoond

SubscriptionResponse

	Veld	**Aantal**	**Uitleg**
SubscriptionResponse	success	(1)	Boolean-waarde die aangeeft of het haltesysteem succesvol aangemeld is bij het distributiesysteem (samenvatting van onderstaande status)
	status	(1)	Een van de volgende statussen[1] (zie ook hoofdstuk ‘Flows’): *REQUEST_INVALID* - Het aanmeldbericht voldoet niet aan de specificatie daarvoor: een of meerdere velden is ongeldig. In het dashboard is terug te vinden welk veld het betreft. *STOP_INVALID* - Het aanmeldbericht heeft de juiste opbouw, maar het haltesysteem meldt zich aan op een bij het distributiesysteem onbekende stopplace- of quaycode (op basis van bron CHB) *AUTHORISATION_REQUIRED* - Het aanmeldbericht is juist, maar de betreffende combinatie van e-mailadres en systeem identificatie zijn nog niet geautoriseerd. *PLANNING_SENT* - Indien het haltesysteem succesvol is aangemeld en er een planning is wordt deze status verstuurd na het succesvol versturen van de planning. *NO_PLANNING* - Indien de aanmelding succesvol is, maar er geen planning is voor de betreffende quay of stopplace-code wordt deze status verstuurd. *AUTHORISATION_VALIDATED* - Het systeem is door een gebruiker zojuist geautoriseerd en de aanmelding is hiermee voldaan. DATA_CHANGED – Indien er een Quay vervalt of als er een Quay op een stoppunt is bijgekomen waarop een subscription is aangevraagd. Wordt deze status gestuurd. De halte wordt dan geacht de verbinding opnieuw op te bouwen en de nieuwe data te verwerken, zonder dat er een onderbreking in de reisinformatie voorziening is.
	timestamp	(1)	Tijdstip waarop het antwoord op het aanmeldbericht is samengesteld.

[1] Er is in de enumeratie ‘Status’ ruimte gelaten voor eventuele toekomstige uitbreidingen van deze enumeratie. De responses zijn dan ook gegroepeerd op categorie.

Unsubscribe

	Veld	Aantal	Uitleg
Unsubscribe	client_id	(1)	Client Id waarmee systeem is aangemeld
	is_permanent	(1)	*Indien nee:* dit betreft een last-will bericht en het haltesysteem is tijdelijk zijn verbinding met de MQTT broker verloren. *Indien ja:* het systeem wordt buiten dienst genomen of wordt niet meer op de huidige manier ingezet. Autorisatiegegevens worden door het distributiesysteem gewist.
	timestamp	(0..1)	Tijdstip waarop het bericht is samengesteld.

Container

		Veld	Aantal	Uitleg
Container	passing_times		(0..1)	Een container object. Kan een tabel met vertrekkende ritten bevatten (PassingTimes)
	PassingTimes	pass_time_hash	(1..n)	(Zie ook "*Standaard berichtafhandeling"*) Voor elke vertrekkende rit wordt een unieke hash berekend dat het haltesysteem kan gebruiken om de ritpassage te identificeren. Dit veld wordt altijd gevuld.
		target_arrival_time	(0..n)	Aankomsttijd volgens dienstregeling van de rit op de halte. Dit kan gebruikt worden om de vertraging ten opzichte van de dienstregeling te berekenen en tonen (bijvoorbeeld "11:10 +3"). De waarde betreft een unix timestamp, oftewel het aantal seconden sinds 1 januari 1970, in UTC. Afnemend haltesysteem is verantwoordelijk om deze informatie te vertalen naar de huidige tijd in tijdzone "Europe/Amsterdam" (tenzij anders gespecificeerd). Dit veld wordt niet verstuurd voor een beginhalte.
		target_departure_time	(0..n)	Vertrektijd volgens dienstregeling van de rit op de halte. Dit betreft een timestamp (zie hierboven). Dit veld wordt niet verstuurd voor een eindhalte.
		expected_arrival_time	(1..n)	De actuele prognose van de aankomsttijd op de betreffende halte van deze rit. Dit betreft een timestamp (zie hierboven), dit veld is altijd leeg voor beginhaltes. Het veld is altijd gevuld met de best bekende aankomsttijd.
		expected_departure_time	(1..n)	De actuele prognose van de vertrektijd op de betreffende halte van deze rit. Dit betreft een timestamp (zie hierboven), dit veld kan niet worden gefilterd, en is altijd leeg voor eindhaltes. Het veld is altijd gevuld met de best bekende vertrektijd.
		number_of_coaches	(1..n)	Het aantal rijtuigen of gekoppelde voertuigen dat deze rit uitvoert.
		trip_stop_status	(1..n)	Ritstatus voor het vertrek op de betreffende halte van de rit. Waarde is op basis van de enumeratie TripStopStatus[2], met als mogelijke waarden: - PLANNED - de rit is gepland - CANCELLED - de rit is opgeheven door een probleem van tijdelijke aard en wordt niet uitgevoerd. Een ritgebonden bericht wordt getoond om de oorzaak van de uitval aan te geven. - DRIVING - de rit is begonnen en er is een actuele status bekend, maar de betreffende halte is nog niet bereikt

[2] Deze enumeratie lijkt op de TripStopStatus enumeratie van Koppelvlak 7/8, maar met 'CANCELLED' ipv CANCEL.

				- ARRIVED - de rit is aangekomen op de halte[3]. - PASSED - de rit is de halte gepasseerd - UNKNOWN - van het voertuig op de rit is geen actuele status bekend
		transport_type	(1..n)	Modaliteit voor betreffende vertrek, bijvoorbeeld om een symbool te tonen. Waarde is op basis van de BISON enumeratie TransportType, met als mogelijke waarden: - BUS - TRAM - METRO - TRAIN - BOAT
		wheelchair_accessible	(1..n)	Geeft aan of het voertuig op de rit rolstoeltoegankelijk is (bijvoorbeeld om een symbool te tonen)
		is_timingstop	(1..n)	Geeft aan of het betreffende vertrek een tijdhalte voor de betreffende rit is.
		stop_code	(1..n)	Quay van het betreffende vertrek. Van belang voor haltesystemen die geabonneerd zijn op meerdere quays en onderscheid willen maken.
		destinations	(1..n)	Tabel met bestemming- en detailbesteminggegevens van de rit.
	Destination	destination_name	(1..n)	Bestemming van de rit. Dit wordt door het distributiesysteem gevuld met de meest toepasselijke bestemming (qua lengte) voor het haltesysteem aan de hand van de aangeleverde maximum lengte in het Subscribe bericht. Of het wordt gevuld met een lijst met alle lengtes van de bestemming die bekend zijn in het distributiesysteem indien het haltesysteem heeft aangegeven zelf de best passende bestemming te kiezen. (SELF_DETERMINING).
		destination_detail	(0..n)	Idem voor de detail bestemmingen.
		show_cancelled_trip	(0..1)*	Geeft aan of een vervallen rit wel of niet getoond moet worden. Mogelijke waardes: - TRUE: Vervallen rit wordt wel getoond, eventueel wordt een verklarende vrije tekst getoond. - FALSE: de vervallen rit wordt niet getoond. Er wordt een verklarende vrije tekst getoond..
		block_code	(0..1)*	Geeft de omloop waarbinnen de rit wordt uitgevoerd, zoals opgenomen in de planning van de betreffende rit. Wordt o.a. gebruikt door een haltesysteem met Vecom detectie.
		occupancy	(0..1)*	Actuele bezettingsgraad van het voertuig (0 – 100%).

[3] Deze status wordt herhaald gestuurd zolang het voertuig op de halte staat.

		line_public_number	(1..n)	Lijnnummer of andere identificatie van de lijn zoals bij de klant bekend (bijvoorbeeld 9K, 251, 12 of N1)
		side_code	(1..n)	Perronaanduiding (bijvoorbeeld A, G, etc)
		line_direction	(1..n)	Numerieke waarde die de richting van de betreffende rit aangeeft. Dit kan door het haltesysteem worden gebruikt om een bepaalde lijn/richting combinatie maar een beperkt aantal keer te tonen.
		line_color	(1..n)	Lijnkleur indien bekend (als achtergrond indien gecombineerd met tekst)
		line_text_color	(1..n)	Lijnkleur indien bekend (als voorgrond indien gecombineerd met tekst)
		line_icon	(1..n)	Lijnicoon om te tonen indien bekend. Bevat, conform de specificatie van KV1, een URL naar een op internet bereikbare afbeelding.
		destination_color	(1..n)	Bestemmingskleur indien bekend (als achtergrond indien gecombineerd met tekst)
		destination_text_color	(1..n)	Bestemmingskleur indien bekend (als voorgrond indien gecombineerd met tekst)
		destination_icon	(1..n)	Bestemmingsicoon om te tonen indien bekend. Bevat, conform de specificatie van KV1, een URL naar een op internet bereikbare afbeelding.
		generated_timestamp	(1..n)	Het tijdstip waarop de reisinformatie over de betreffende rit gegenereerd is. Kan gebruikt worden om te verifiëren dat opeenvolgende berichten in de juiste volgorde worden verwerkt.
		journey_number	(1..n)	Ritnummer zoals bekent bij de vervoerder.
	general_messages			Een container object. Kan een tabel met vrije teksten bevatten (GeneralMessage)
	GeneralMessage	message_hash	(1..n)	(Zie ook "*Standaard berichtafhandeling"*) Voor elke vrije tekst/bericht wordt een unieke hash berekend dat het haltesysteem kan gebruiken om het bericht te identificeren. Dit veld wordt altijd gevuld.
		generalmessage_type	(1..n)	Het type vrije tekst. Waarde is op basis van de BISON enumeratie MessageType, met als mogelijke waarden: - GENERAL. De vrije tekst wordt getoond, afhankelijk van de technische mogelijkheden van het display, zoals in de weergaverichtlijnen beschreven - OVERRULE. Er worden geen ritregels getoond en de vrije tekst wordt centraal op het display getoond. - BLANC. Het display toont geen informatie.
		message_content	(0..n)	Inhoud van de vrije tekst indien bekend.

		message_start_time	(0..n)	Begintijd vanaf wanneer de vrije tekst getoond moet worden (dit is een timestamp, zie uitleg in bericht 'Container')
		message_end_time	(0..n)	Eindtijd tot wanneer de vrije tekst getoond moet worden (dit is een timestamp, zie uitleg in bericht 'Container'). Indien geen eindtijd wordt gespecificeerd moet het bericht getoond worden tot het wordt verwijderd.
		show_overview_display	(0..1)*	Geeft aan of de tekst ook getoond moet worden op een overzichtsdisplay. Mogelijke waardes: TRUE: de tekst moet ook op een overzichtdisplay worden getoond. FALSE: de tekst moet niet op een overzichtdisplay getoond worden. ONLY: de tekst moet alleen op een overzichtdisplay getoond worden en niet op een haltedisplay.
		message_title	(0..1)*	Titel van de vrije tekst. Afhankelijk van de mogelijkheden van het display, wordt de titel apart (en groter) getoond of wordt de titel opgenomen in de tekst.
		message_priority	(0..1)*	Geeft een indeling naar de inhoud van de tekst. Het geeft daarmee het belang van de tekst aan. Mogelijke waardes: CALAMITY: tekst is gerelateerd aan een acute en ongeplande verstoring waar veel reizigers hinder van ondervinden. PTPROCESS: mededelingen als gevolg van veelvoorkomende afwijkende situaties, zoals omleidingen, vertragingen enz. COMMERCIAL: Aankondigingen van wijzigingen in de toekomst, zoals nieuwe dienstregelingen, enz. MISC: Overige mededelingen.
	general_messages_remove			Een container object. Kan een tabel met vrije teksten bevatten die verwijderd moeten worden (GeneralMessageRemove)
	GeneralMes	message_hash	(1..n)	(Zie ook "*Standaard berichtafhandeling"*) Voor elke vrije tekst/bericht wordt een unieke hash berekend dat het haltesysteem kan gebruiken om het bericht te identificeren. Dit veld wordt altijd gevuld.
		generated_timestamp	(1..n)	Het tijdstip waarop het verwijderbericht is gegenereerd.
	public_names			Een container object. Kan een tabel met publieke namen van de halte bevatten volgens het CHB (PublicName).
	PublicName	stop_code	(0..n)	Quay waarover het haltesysteem informeert.
		public_name_place	(0..n)	Publieke naam van de place zoals bekend is bij het CHB
		public_name_stop_place	(0..n)	Publieke naam van de stopplace zoals bekend is bij het

				CHB
		public_name_quay	(0..n)	Publieke naam van de Quay zoals bekend is bij het CHB

* Deze velden zijn opgenomen voor toekomstig gebruik.

Indien bij de subscription bij destination_determination is aangegeven dat er gebruik gemaakt wordt van SELF_DETERMINING dan wordt de destinations als volgt gevuld per ritregel.
destination_name = [DestinationName50, DestinationName30, DestinationName24, DestinationName19, DestinationName16]
destination_detail = [“”, “”, DestinationDetail24, DestinationDetail19, DestinationDetail16]
De eerste 2 waarden van destination_detail zullen dus leeg zijn omdat deze gegevens niet via de huidige KV7/8 aangeleverd worden.

PublicName wordt gevuld in het eerste containerbericht na het succsesvol aanmelden van een haltesysteem. Dubbele waarden worden er uitgefilterd. Bijvoorbeeld:
Als een haltesysteem zich aanmeldt voor data van ‘NL:S:50000120’, dan wordt de volgende data verstuurd in public_names.

public_name_place = [Utrecht centraal];
public_name_stop_place = [CS Jaarbeurszijde];
public_name_quay = [Halte C1, Halte C2, Halte C3, Halte C4, Halte C5, Halte C6, Halte C7, Halte C8, Halte C9, Halte C10];
stop_code=[NL:Q:50000120,NL:Q:50000121,NL:Q:50000122,NL:Q:50000123,NL:Q:50000124,NL:Q:50000125,NL:Q:50000126,NL:Q:50000127,NL:Q:50000128,NL:Q:50000129];

Als een haltesysteem zich aanmeldt voor data van ‘NL:Q:50000120’ dan wordt de volgende data verstuurd in public_names.
public_name_place = [Utrecht centraal];
public_name_stop_place = [CS Jaarbeurszijde];
public_name_quay = [Halte C1];
stop_code=[NL:Q:50000120];

Berichten Distributiesysteem of Haltesysteem en Dashboardsysteem

SystemStatus

Dit bericht wordt generiek gebruikt door zowel distributiesysteem als haltesystemen om te communiceren naar een dashboardsysteem over de status van een systeem en de onderdelen daarvan. De status kan daarbij uitgedrukt worden in een van twee types: meetwaardes (Metric) of logberichten (LogMessage). Indien de informatie betrekking heeft op een haltesysteem, wordt dit in het client id duidelijk.

		Veld	**Aantal**	**Uitleg**
SystemStatus	metrics		(0..n)	Actuele meetwaardes die gecommuniceerd moeten worden. Meetwaardes worden met regelmatige intervallen gepubliceerd door het metende systeem.
	Metric	Client_id	(1)	Client Id waarmee systeem is aangemeld
		component	(1)	Naam van het betreffende component zoals gedefinieerd door de opdrachtgever (bijvoorbeeld “Behuizing” of “Display” voor haltesystemen of “Planning” voor een distributiesysteem). Toegestane tekens zijn letters en eventueel een lage streep (A-Z, a-z en _)
		component_index	(0..1)	Identificatie van het betreffende component in het systeem voor het geval er meerdere dezelfde componenten zijn. Optioneel, indien er nooit een tweede component is kan dit leeg worden gelaten.
		property	(1)	Eigenschap van het betreffende component dat wordt gemeten (bijvoorbeeld “deur”, “temperatuur”, “signaalsterkte” voor een haltesysteem of “ritten” of “berichten” voor een distributiesysteem). Toegestane tekens zijn letters en eventueel een lage streep (A-Z, a-z en _).
		value	(1)	Numerieke waarde voor de de betreffende eigenschap van het component (boolean waardes, goed/fout of verbonden/verbroken kunnen als 0/1 worden gecommuniceerd). Voor een betreffende combinatie ‘component’ en ‘property’ moet de eenheid waarin de waarde gemeten wordt altijd hetzelfde zijn.
		unit	(1)	De eenheid van de waarde. Het mag geen newlines/enters bevatten, verder zijn er geen beperkingen (geldige UTF-8 content).
		timestamp_begin	(1)	Het begintijdstip van de periode waarover de waarde is gemeten. (dit is een timestamp, zie uitleg in bericht ‘Container’)
		timestamp_end	(1)	Het eindtijdstip van de periode waarover de waarde is gemeten. Indien de meetwaarde een momentane waarde is, blijft dit veld leeg (dit is een timestamp, zie uitleg in bericht ‘Container’)
	logs		(0..n)	Eventuele logberichten die gecommuniceerd moeten worden. (LogMessage)

	LogMessage	client_id	(1)	Client Id waarmee systeem is aangemeld
		type	(1)	Een van de volgende waardes van de enumeratie 'StatusType': - *ERROR*: de boodschap betreft de statusovergang naar een foutsituatie - *WARNING*: de boodschap betreft de statusovergang naar een waarschuwing - *OK*: de boodschao betreft statusovergang van een ERROR of WARNING terug naar een geaccepteerde waarde. In dit geval wordt de duur van de storing of waarschuwing in seconden meegegeven. - *LOG:* de boodschap betreft een andere, meer algemene, melding die geen statusovergang betreft.
		code	(1)	Een numerieke code die het type gebeurtenis aangeeft waarover een logbericht wordt gecommuniceerd[4].
		message	(1)	De tekstuele boodschap behorende bij de geregistreerde gebeurtenis, deze wordt in een dashboard aan de gebruiker gepresenteerd. Het bericht mag geen newlines/enters bevatten, verder zijn er geen beperkingen (geldige UTF-8 content).
		duration	(0..1)	De duur van een storing (ERROR) of waarschuwing (WARNING) in seconden. Deze waarde wordt alleen gevuld bij een boodschap van het type OK.
		timestamp	(0..1)	Het tijdstip waarop het bericht gegenereerd is (dit is een timestamp, zie uitleg in bericht 'Container')

[4] Deze code moet uniek zijn en vergemakkelijkt het processen van logberichten in de dashboards

Berichten tussen Halte- en Dashboardsysteem

Begrippen en regels

Bij de berichtenstroom tussen halte- en dashboardsystemen gelden de volgende begrippen en regels:

- Een haltesysteem (in de praktijk ook vaak display genoemd) bestaat uit componenten. Voorbeeld: behuizing, scherm, software, …
- Een component heeft eigenschappen. Voorbeeld: een behuizing heeft een deur, een scherm heeft een aantal defecte LED's, …
- Eigenschappen hebben een unieke code[5].
- Eigenschappen hebben een waarde. Voorbeeld: een deur is open, het aantal defecte LED's is 4.
- Sommige eigenschappen kennen conditie-overgangen. Deze treden op als de waarde een grens overschreidt. Voorbeeld: een deur gaat van dicht naar open, het aantal defecte LED's gaat van 4 naar 5, waarbij 5 als grens is gedefinieerd.
- De waardes van alle eigenschappen tezamen vormen de status van het haltesysteem.

Bij de aanschaf van een haltesysteem wordt overeengekomen uit welke componenten een haltesysteem bestaat en welke eigenschappen die componenten hebben. Ook wordt van elke conditie-overgang het statustype bepaald (ERROR, WARNING of OK)
Het haltesysteem is verantwoordelijk voor het publiceren van de overeengekomen conditie-overgangen naar de dashboards zodat gebruikers de status van de haltesystemen kunnen volgen, zonder dat er regelmatig statusinformatie moet worden opgevraagd. Indien een eigenschap van een component van een haltesysteem een conditie-overgang ondergaat, wordt een 'SystemStatus' bericht op het topic van het haltesysteem gepubliceerd. Indien een dashboardsysteem meer informatie nodig heeft kan de gehele status van een haltesysteem worden opgevraagd. Het is binnen het koppelvlak niet mogelijk om de grenswaarden voor conditie-overgangen in te stellen. Dit wordt verondersteld onderdeel uit te maken van een configuratiebestand op het haltesysteem.

- Het haltesysteem **kan** een logboodschap versturen als de waarde van een eigenschap verandert.
- Het haltesysteem verstuurt **altijd** een logboodschap bij een overeengekomen conditie-overgang.
- Logboodschappen bij conditie-overgangen hebben een type ERROR, WARNING of OK.
- Logboodschappen van het type ERROR of WARNING, van een eigenschap (=code) worden **altijd** gevolgd door een logboodschap van het type OK van diezelfde eigenschap. Voorbeeld: deur open (WARNING) moet altijd gevolgd worden door een deur dicht (OK).
- Bij elke logboodschap van het type OK berekent het haltesysteem de duur van de storing of waarschuwing en stuurt dat mee met de logboodschap.

Informatieverzoeken

Vanuit de dashboardsystemen kunnen vier verschillende informatieverzoeken naar een haltesysteem worden verstuurd. Het haltesysteem reageert hierop met een antwoord. De vier verzoeken zijn voor het ophalen van:

- De gegevens over het systeem:
  gedefinieerd in naam/waarde combinaties. Bijvoorbeeld: schermtype = LED, of aantal regels = 4.
- De actuele status:
  lijst met alle eigenschappen waarvoor een conditie-overgang is gedefinieerd, met de actuele waarde van de eigenschap plus het type van de laatste logboodschap.

[5] Deze codes worden beheerd door OV-data en zijn uniek over de DRIS implementaties heen.

- De reisinformatie in het haltesysteem:
  informatie over actuele en toekomstige ritten, alsmede alle actuele en toekomstige vrije teksten.
- De tekst die op het scherm staat:
  informatie die getoond wordt op het scherm of de schermen (snapshot).

Bestandsuitwisseling

Het centrale distributiesysteem heeft een FTP server beschikbaar waarmee bestanden kunnen worden uitgewisseld tussen dashboard en haltesystemen. Een dashboardsysteem kan een configuratiebestand klaarzetten op de FTP server en stuurt vervolgens een bericht naar één of meer haltesystemen dat het bestand kan worden opgehaald.
Andersom kan een haltesysteem (regelmatig) een tracebestand klaarzetten op de FTP server en een bericht sturen naar een dashboardsysteem. Deze mogelijkheid om tracebestanden te maken in het haltesysteem en deze naar een dashboardsysteem te versturen is alleen bedoeld voor incidentele analyse. Het is uitdrukkelijk niet de bedoeling dat meerdere haltesystemen gedurende langere tijd gebruik maken van deze faciliteit.
De FTP servers tussen de beide clusters worden niet gesynchroniseerd. Een haltesysteem kan op enig moment communiceren met één van beide clusters en het is de verantwoordelijkheid van de dashboardsystemen om de configuratiebestanden op beide FTP servers te plaatsen. Evenzo zet een haltesysteem een tracebestand neer op de cluster waarmee het op dat moment communiceert en het is de verantwoordelijkheid van de dashboardsystemen om te onderzoeken op welke cluster een tracebestand staat.

InfoRequest

	Veld	**Aantal**	**Uitleg**
InfoRequest	client_id	(1)	Client Id waarmee het (dashboard)systeem is aangemeld.
	request_type	(1)	Een van de volgende waardes van de enumeratie ‘RequestType: - *SYSTEM_INFO*: het dashboard vraagt de algemene informatie van het haltesysteem op. - *STATUS*: het dashboard vraagt de actuele waarde van de eigenschappen op. - *TRAVEL_INFO*: het dashboard vraagt de informatie op van actuele en toekomstige ritten en vrije teksten. - *SCREEN_CONTENT:* het commando vraagt wat getoond wordt op het scherm of op de schermen.

SystemInfo

Met dit bericht stuurt een haltesysteem de lijst met overeengekomen[6] informatie-elementen in de vorm van naam/waarde paren. Het bericht wordt verstuurd in antwoord op een verzoek van een dashboardsysteem.

SystemInfo	client_id		(1)	Client Id waarmee het haltesysteem is aangemeld
	system_info_lines		(1…n)	Lijst met met Informatieregels over het haltesysteem. Elke regel bestaat uit een naam plus een waarde.
	SystemInfoLine	Name	(1)	De, niet gestandaardiseerde, naam van het informatie-element.
		value_type	(1)	Geeft aan van welk technisch type de waarde is (string, integer, double, bool, byte). Hiermee kan het ontvangende dashboardsysteem het juiste veld uitlezen.
		string_value	(0..1)	Dit veld wordt gevuld met de actuele waarde van de eigenschap als de waarde in tekst wordt gegeven.
		int_value	(0..1)	Dit veld wordt gevuld met de actuele waarde van de eigenschap als de waarde numeriek is.
		double_value	(0..1)	Dit veld wordt gevuld met de actuele waarde van de eigenschap als de waarde double is.
		bool_value	(0..1)	Dit veld wordt gevuld met de actuele waarde van de eigenschap als de waarde boolean is.
		bytes_value	(0..1)	Dit veld wordt gevuld met de actuele waarde van de eigenschap als de waarde in bytes is.

[6] Bij de aanschaf van het haltesysteem wordt een lijst met gegevens overeengekomen.

StatusOverview

Met dit bericht stuurt een haltesysteem een lijst met de actuele waardes van van alle overeengekomen eigenschappen van het haltesysteem plus het type van de laatste Logboodschap die is verstuurd (LogMessage). Tesamen vormen deze waardes de actuele status van het haltesysteem.

StatusOverview	client_id		(1)	Client Id waarmee (halte)systeem is aangemeld
	status_overview_lines		(1…n)	Lijst met met statusregels.
	StatusOverviewLine	component	(1)	De door DOVA gestandaardiseerde naam van een deel van een haltesysteem. De componentnamen worden bij aanschaf van de haltesystemen vastgelegd.
		property	(1)	De naam van de componenteigenschap waarover gerapporteerd wordt.
		value_type	(1)	Geeft aan welk van technisch type de waarde is (string, integer, double, bool, byte). Hiermee kan het ontvangende dashboardsysteem het juiste veld uitlezen.
		string_value	(0..1)	Dit veld wordt gevuld met de actuele waarde van de eigenschap als de waarde in tekst wordt gegeven.
		int_value	(0..1)	Dit veld wordt gevuld met de actuele waarde van de eigenschap als de waarde integer is.
		double_value	(0..1)	Dit veld wordt gevuld met de actuele waarde van de eigenschap als de waarde double is.
		bool_value	(0..1)	Dit veld wordt gevuld met de actuele waarde van de eigenschap als de waarde boolean is.
		bytes_value	(0..1)	Dit veld wordt gevuld met de actuele waarde van de eigenschap als de waarde in bytes is.
		status	(1)	Het type van de laatst verstuurde logboodschap (ERROR, WARNING, OK, LOG). De enumeratie is beschreven bij de LogMessage.

TravelInfo

Met dit bericht stuurt het haltesysteem alle vertrektijden en vrije teksten die het heeft.

	Veld	**Aantal**	**Uitleg**
TravelInfo	client_id	(1)	Client Id waarmee het (halte)systeem is aangemeld.
	Travel_info_content	(1)	Dit bericht ‘Container’ is eerder gedocumenteerd. Het haltesysteem moet alle op dat moment bekende ritten en vrije teksten in het bericht opnemen en publiceren, zodat gecontroleerd kan worden of de juiste informatie opgeslagen is.

ScreenContentResponse

Met dit bericht stuurt het haltesysteem de scherminhoud inclusief eventuele op dat moment niet zichtbare inhoud, bijvoorbeeld bij alterneren of als meerdere blokken vrije tekst worden getoond. Indien op het haltesysteem meerdere schermen zijn aangesloten die dezelfde inhoud tonen, wordt de inhoud 1 keer verstuurd. Indien er meerdere schermen op het haltesysteem zijn aangesloten die verschillende inhoud tonen, moet van iedere set schermen die dezelfde inhoud tonen één scherminhoud gestuurd worden.

Dit bericht moet eenmalig gestuurd worden als antwoord op een InfoRequest van het type SCREEN_CONTENT en vervolgens steeds als één van de scherminhouden verandert[7], voor een periode van 5 minuten na het initiele antwoord.

	Veld		**Aantal**	**Uitleg**
ScreenContentResponse	client_id		(1)	Client Id waarmee het (halte)systeem is aangemeld.
	screen_contents		(0..n)	
	ScreenContent	screen_index	(0..1)	Indien een haltesysteem meerdere schermen heeft die verschillende informatie tonen, is dit het volgnummer van het betreffende scherm[8]. Indien het haltesysteem maar uit een scherm bestaat of uit meerdere schermen die dezelfde informatie tonen, mag dit veld leegblijven.
		content_type	(1)	Een van de volgende formaten die aangeven hoe het veld 'content' is gevuld: - *TEXT*: De geformatteerde tekst zoals te zien is op het display (op het moment van opname, inclusief het alternerende deel van de tekst) - *IMAGE_PNG*: Een afbeelding in Portable Network Graphics (PNG) formaat (op het moment van opname, inclusief het alternerende deel van de tekst) - *IMAGE_GIF*: Een afbeelding in het Graphics Interchange Format (GIF) formaat. Hierbij is het mogelijk meerdere (bewegende) frames op te nemen om alternerende teksten weer te kunnen geven. De leverancier van het haltesysteem kiest aan de hand van de eisen van opdrachtgever en type scherm het meest geschikte formaat.
		content	(1)	Indien type = TEXT, de ruwe, geformateerde tekst. Indien type = IMAGE_PNG of IMAGE_GIF, base64 encoded data van de afbeelding.
		timestamp	(1)	Tijdstip waarop dit bericht is samengesteld door het haltesysteem (dit is een timestamp, zie uitleg in bericht 'Container')

[7] Let op dat alterneren, of het tonen van een ander vrije tekst blok geen verandering is die een nieuw bericht tot gevolg heeft, omdat de scherminhoud wordt verstuurd inclusief de inhoud die niet zichtbaar is door alterneren of meerdere blokken vrije tekst.
[8] Scherm 1 toont de eerst vertrekkende ritten, scherm 2 de volgende enz.

FileAvailable

Dashboardsystemen kunnen een configuratiebestand klaarzetten voor een haltesysteem. Op de centrale distributiecluster is daartoe een FTP server beschikbaar. Andersom kunnen haltesystemen tracebestanden op de FTP server klaarzetten voor dashboardsystemen. Het afnemende systeem kan geinformeerd worden dat er een bestand beschikbaar is met het hier beschreven bericht.

	Veld	**Aantal**	**Uitleg**
FileAvailable	client_id	(1)	Client Id waarmee het (dashboard of halte)systeem is aangemeld.
	file_name	(1)	Naam van het bestand dat klaar staat op de FTP server.

MQTT

Algemeen
Om data te kunnen versturen en ontvangen, moeten alle systemen (Distributie-, Halte- en Dashboardsystemen) verbonden worden met de centrale MQTT broker, die werkt conform MQTT versie 5.

Verbinden
Bij het verbinden van de clientsystemen met de MQTT broker zijn de volgende instellingen van belang:

- Keep Alive
  De Keep Alive periode moet instelbaar zijn in de clientsystemen met een defaultwaarde van 15 seconden voor Distributie- en Dashboardsystemen en 60 seconden voor Haltesystemen.

- Client id
  Voor de verbinding met de broker is een naam nodig. De naam wordt samengesteld uit de drie velden van Client Id gescheiden door een '_' (Underscore): de volgorde is als volgt.
  - SubscriberOwnerCode
  - Type (0 = Distributiefunctie, 1 = Dashboardfunctie, 2 = Haltefunctie)
  - Serialnumber

  Een voorbeeld van een client id is: DOVA_1_1 of SURTRONIC_2_42741

  Een client id dient uniek te zijn over alle systemen heen. Indien het client id wijzigt zal de autorisatieprocedure opnieuw moeten worden doorlopen.

- Clean Start
  True voor Distributie- en Haltesystemen en False voor Dashboardsystemen.
  Omdat een distributiesysteem alleen informatie stuurt, is een sessie met geschiedenis niet relevant. Een Haltesysteem krijgt bij aanmelden (initieel en na verlies van de verbinding) alle informatie opnieuw opgestuurd. Een sessie met geschiedenis is daarom niet relevant.
  Een Dashboardsysteem moet echter altijd een complete historie bijhouden van de aangesloten Halte- en Distributiesystemen en moet daarom alle informatie die gestuurd is, ontvangen. Een sessie met (lange) geschiedenis is daarom van belang. Expiry Interval voor het Dashboard wordt op : 604800 Sec gezet (1Week)

- Last Will
  Bij verlies van de verbinding tussen een clientsysteem en de Broker moet altijd een Last Will worden uitgezonden, zodat het verbindingsverlies bekend wordt in de Dashboardsystemen.
  Voor Distributiesystemen moet op topic *unsubscribe/<version>/0/<SubscriberOwnerCode>/<Serialnumber>* een Unsubscribebericht geconfigureerd staan dat aangeeft dat de verbinding is verbroken.
  Voor Haltesystemen moet op topic *unsubscribe/<version>/2/<SubscriberOwnerCode>/<Serialnumber>* een Unsubscribe bericht geconfigureerd staan met als enige waarde "is_permanent" = false (dit bericht kan dus eenmalig tijdens initialisatie worden aangemaakt). Let op dat dit bericht ook door het haltesysteem moet worden verstuurd als de software of het haltesysteem moet herstarten om welke reden dan ook.
  Voor Dashboardsystemen moet op topic *unsubscribe/<version>/1/<SubscriberOwnerCode>/<Serialnumber>* een Unsubscribebericht geconfigureerd staan dat aangeeft dat de verbinding is verbroken.

- VersionNr
  Om in de toekomst verschillende versies te kunnen ondersteunen wordt er per topic een versienummer opgenomen. Versienummers bestaan uit één getal en geen subgetallen. In deze versie van het document wordt versie 1 gebruikt. Er wordt maar één getal gebruikt omdat er verondersteld wordt dat het altijd om een significante wijziging gaat bijvoorbeeld als er een veld verwijderd of toegevoegd wordt. Minder significante wijzigingen, zoals bijvoorbeeld een documentaanpassing, worden met een subnummer aangegeven.

Topics

Per topic wordt exact één type bericht gepubliceerd zodat halte-, distributie en dashboardsystemen weten welke berichten verwacht kunnen worden.

subscribe/<Version>/<Type>/<SubscriberOwnerCode>/<Serialnumber>
Publicatie: Individueel haltesysteem
Abonnee: Distributiesysteem, Dashboardsystemen
Bericht: Subscribe
QoS: Exactly once (2)
Retain: false
In het aanmeldproces wordt dit topic gebruikt door haltesystemen om aanmeldinformatie op te publiceren. Hierdoor weten het distributiesysteem en de dashboardsystemen welke haltesystemen op dat moment actief zijn en op welke topics gepubliceerd moet worden om te communiceren met het haltesysteem. Ook krijgt het distributiesysteem eventuele stuurparameters zoals bijvoorbeeld filters, meegestuurd.

unsubscribe/<Version>/<Type>/<SubscriberOwnerCode>/<Serialnumber>
Publicatie: Distributiesysteem, Haltesysteem
Abonnee: Distributiesysteem, Dashboardsystemen en Haltesysteem
Bericht: Unsubscribe
QoS: Exactly once (2)
Retain: false
Dit topic wordt gebruikt voor zowel het last-will bericht als permanente afmeldberichten.

travel_information/<Version>/<Type>/<SubscriberOwnerCode>/<Serialnumber>
Publicatie: Distributiesysteem
Abonnee: Individueel haltesysteem
Bericht: Container
QoS: Atleast once (1)
Retain: false
Reisinformatieberichten zoals hierboven omschreven worden op dit topic gepubliceerd. Dit kunnen zowel passeertijden als vrije teksten zijn. Passeertijden kunnen op korte termijn (realtime) of langere termijn (planning) verstuurd worden.

De QoS op dit topic is lager dan andere topics omdat de reisinformatieberichten op twee manieren van elkaar te onderscheiden zijn (door middel van het veld ‘pass_time_hash’ en het optioneel te ontvangen veld ‘generated_timestamp’). Hierdoor is er dus geen nadeel als hetzelfde bericht meerdere keren wordt ontvangen door een haltesysteem. Dit heeft daarnaast ook performance voordelen.

subscription_response/<Version>/<Type>/<SubscriberOwnerCode>/<Serialnumber>
Publicatie: Distributiesysteem
Abonnee: Individueel haltesysteem, dashboardsystemen
Bericht: SubscriptionResponse
QoS: Exactly once (2)
Retain: false
Resultaten en acties met betrekking tot de subscription worden op dit topic gepubliceerd door het distributiesysteem. Haltesystemen moeten op dit topic abonneren om de status te ontvangen van een aanmelding en eventuele vervolg acties uit te voeren. Er wordt voor elk aanmeldbericht op

subscribe/<Version>/<Type>/<SubscriberOwnerCode>/<Serialnumber> een antwoord verstuurd op dit topic. De dashboardsystemen kunnen zich op dit topic abonneren, zodat ze het subscriptionproces kunnen volgen en vastleggen.

systemstatus/<Version>/<Type>/<SubscriberOwnerCode>/<serialnumber>
Publicatie: Individueel Haltesysteem, Distributiesysteem, individueel Dashboardsysteem
Abonnee: Dashboardsystemen (autorisatie wordt afgedwongen op basis van SubscriberOwnerCode)
Bericht: SystemStatus
QoS: Atleast once (1)
Retain: false
Op dit topic publiceren alle soorten systemen logboodschappen en meetwaarden. Met de logboodschappen kan in een dashboardsysteem de status van een systeem gevolgd worden en de meetwaarden geven (statistische) informatie over de werking van het systeem.

inforequest/<Version>/<Type>/<SubscriberOwnerCode>/<serialnumber>
Publicatie: individueel Dashboardsysteem
Abonnee: individueel haltesysteem.
Bericht: InfoRequest
QoS: Atleast once (1)
Retain: false
Op dit topic publiceren de dashboardsystemen hun verzoek om informatie van een haltesysteem.

Systeminfo/<Version>/<Type>/<SubscriberOwnerCode>/<serialnumber>
Publicatie: individueel haltesysteem
Abonnee: dashboardsysteem
Bericht: SystemInfo
QoS: Atleast once (1)
Retain: false
Op dit topic publiceert een haltesysteem de systeeminformatie in antwoord op het verzoek van een dashboardsysteem.

Statusoverview/<Version>/<Type>/<SubscriberOwnerCode>/<serialnumber>
Publicatie: individueel haltesysteem
Abonnee: dashboardsysteem
Bericht: StatusOverview
QoS: Atleast once (1)
Retain: false
Op dit topic publiceert een haltesysteem het statusoverzicht in antwoord op het verzoek van een dashboardsysteem.

travelinfo/<Version>/<Type>/<SubscriberOwnerCode>/<serialnumber>
Publicatie: individueel haltesysteem
Abonnee: dashboardsysteem
Bericht: TravelInfo
QoS: Atleast once (1)
Retain: false
Op dit topic publiceert een haltesysteem de informatie over alle ritten en vrije teksten die het heeft, in antwoord op het verzoek van een dashboardsysteem.
screencontent/<Version>/<Type>/<SubscriberOwnerCode>/<serialnumber>

Publicatie: individueel haltesysteem
Abonnee: dashboardsysteem
Bericht: InfoRequest
QoS: Atleast once (1)
Retain: false
Op dit topic publiceert een haltesysteem gedurende vijf minuten de informatie die op het scherm (de schermen) wordt getoond, in antwoord op het verzoek van een dashboardsysteem.

fileavailable/<Version>/<Type>/<SubscriberOwnerCode>/<serialnumber>
Publicatie: individueel haltesysteem of dashboard
Abonnee: individueel dashboardsysteem of haltesysteem
Bericht: FileAvailable
QoS: Atleast once (1)
Retain: false
Op dit topic geeft een dashboardsysteem aan dat er een configbestand klaar staat voor één of meer haltesystemen. Evenzo geeft een haltesysteem op dit topic aan dat er een tracebestand klaar staat voor één of meer dashboardsystemen.

Een haltesysteem moet zich voor de reisinformatie op vier topics abonneren:

- *travel_information/<Version>/2/<SubscriberOwnerCode>/<Serialnumber>. Het haltesysteem abonneert zich op dit topic met het eigen SubscriberOwnerCode en Serialnumber, zodat het alleen de “eigen” reisinformatie ontvangt.*
- *subscription_response/<Version/2/<SubscriberOwnerCode>/<Serialnumber>. Het haltesysteem abonneert zich op dit topic met het eigen SubscriberOwnerCode en Serialnumber, zodat het alleen de antwoorden ontvangt op de “eigen” aanmeldingen.*
- *unsubscribe/<Version>/2/<SubscriberOwnerCode>/<Serialnumber>: Het haltesysteem abonneert abonneert zich op dit topic met het eigen SubscriberOwnerCode en Serialnumber, zodat het alleen de “eigen” afmeldingen (last will) ontvangt.*
- *unsubscribe/<Version>/0/<SubscriberOwnerCode>/<Serialnumber>: Het haltesysteem abonneert zich op dit topic, waarbij het client id het id is van het distributiesysteem waar het op is aangemeld.*

Voor beheerdoeleinden moet een haltesysteem zich verplicht abonneren op de volgende twee topics:

- *inforequest/<Version>/2/<SubscriberOwnerCode>/<serialnumber> Het haltesysteem abonneert zich op dit topic met het eigen SubscriberOwnerCode en Serialnumber, zodat het alleen de “eigen” informatieverzoeken ontvangt.*
- *fileavailable/<Version>/2/<SubscriberOwnerCode>/<serialnumber> Het haltesysteem abonneert zich op dit topic met het eigen SubscriberOwnerCode en Serialnumber, zodat het alleen de “eigen” configuratiebestanden ontvangt.*

Bijlage 1: mapping Kv7/Kv8 t.o.v. Openkoppelvlak

In de onderstaande tabel wordt de mapping van de Open koppelvlak velden met de KV7 Planning data beschreven.

De ‘pass_time_hash’ wordt gegenereerd uit de velden ‘DataOwnerCode, LocalServiceLevelCode, LinePlanningNumber,JourneyNumber,FortyfyOrderNummer, UserStopCode, UserStopOrderNumber en OperationDate)

Open Koppelvlak Velden	KV7 planning	Default value
pass_time_hash	*Gegenereerde hash*	
target_arrival_time	LocalServiceGroupPasstime.TargetArrivalTime	
target_departure_time	LocalServiceGroupPasstime.TargetDepartureTime	
expected_arrival_time	n.v.t.	0
expected_departure_time	n.v.t.	0
number_of_coaches	n.v.t.	0
trip_stop_status	‘PLANNED’	
transport_type	Line.TransportType	
wheelchair_accessible	LocalServiceGroupPasstime.WheelChairAccessible	
is_timing_stop	LocalServiceGroupPasstime.IsTimingStop	
stop_code	PassengerStopAssignment.QuayCode	
destinations	Destination.DestinationNameXX (Waarbij XX afhankelijk is van de subscription.) en DestinationDetailXX (Waarbij XX afhankelijk is van de subscription.)	
show_cancelled_trip	n.v.t.	True
block_code	n.v.t.	<lege string>
Occupancy	n.v.t.	0
line_public_number	Line.LinePublicNumber	
side_code	LocalServiceGroupPasstime.SideCode	
line_direction	LocalServiceGroupPasstime. LineDirection	
line_color	LocalServiceGroupPasstime.LineDestColor	
line_text_color	LocalServiceGroupPasstime.LineDestTextColor	
line_icon	LocalServiceGroupPasstime.LineDestIcon	
destination_color	Destination.DestColor	
destination_text_color	Destination. DestTextColor	
destination_icon	Destination.DestIcon	
generated_timestamp	*Tijdstip waarop record gegenereerd is.*	

De mapping van de openkoppelvlak velden ten opzichte van KV8 berichten wordt in de onderstaande tabel weergegeven.

De ‘pass_time_hash’ wordt gegenereerd uit de velden ‘DataOwnerCode, LocalServiceLevelCode, LinePlanningNumber,JourneyNumber,FortyfyOrderNummer, UserStopCode, UserStopOrderNumber en OperationDate)

Enkele velden worden gevuld uit de CDD_APPLICATION_INFORMATION database. Let op: de time velden worden gevuld met Unix timestamp. Oftewel het aantal seconden sinds 1 januari 1970 00:00:00 in UTC

Open Koppelvlak Velden	KV8 DATEDPASSTIME	Default value
pass_time_hash	*Gegenereerde hash*	
target_arrival_time	(KV7)LocalServiceGroupPasstime.TargetArrivalTime	
target_departure_time	(KV7)LocalServiceGroupPasstime.TargetDepartureTime	
expected_arrival_time	(KV8)ExpectedArrivalTime	
expected_departure_time	(KV8)ExpectedDepartureTime	
number_of_coaches	(KV8)NumberOfCoaches	
trip_stop_status	(KV8)TripStopStatus	
transport_type	(KV7)Line.TransportType	
wheelchair_accessible	(KV8)WheelChairAccessible	
is_timing_stop	(KV8)IsTimingStop	
stop_code	PassengerStopAssignment.QuayCode	
destinations	Destination.DestinationNameXX (Waarbij XX afhankelijk is van de subscription.) en DestinationDetailXX (Waarbij XX afhankelijk is van de subscription.)	
show_cancelled_trip	(KV8)ShowCancelledTrip	
block_code	n.v.t.	<lege string>
Occupancy	n.v.t.	0
line_public_number	(KV7)Line.LinePublicNumber	
side_code	(KV8)SideCode	
line_direction	(KV8)LineDirection	
line_color	(KV8)LineDestColor	
line_text_color	(KV8)LineDestTextColor	
line_icon	(KV8)LineDestIcon	
destination_color	(KV7)Destination.DestColor	
destination_text_color	(KV7)Destination. DestTextColor	
destination_icon	Destination.DestIcon	
generated_timestamp	*Tijdstip waarop record gegenereerd is.*	

De mapping van de general message is als volgt.

De ‘message_hash’ wordt gegenereerd uit de velden ‘DataOwnerCode,MessageCodeDate,MessageCodeNumber,TimingPointDataOwnerCode,TimingPointCode)

Let op: de time velden worden gevuld met unix timestamp. Oftewel het aantal seconden sinds 1 januari 1970 00:00:00 in UTC

Open Koppelvlak Velden	KV8 Generalmessage	Default value
message_hash	*Gegenereerde hash*	
generalmessage_type	KV8.MessageType	
message_content	KV8.MessageContent	
message_start_time	KV8.MessageStartTime	
message_end_time	KV8.MessageEndTime	Max unixtimestamp als leeg aangeleverd
generated_timestamp	*Tijdstip waarop record gegenereerd is.*	
show_overview_display	n.v.t.	True
message_title	n.v.t.	<lege string>
message_priority	Altijd CALAMITY	

```
// Open DRIS versie 3.3

message ClientId {
	enum SubscriberType {
		DISTRIBUTIESYSTEEM = 0;
		DASHBOARDSYSTEEM = 1;
		HALTESYSTEEM = 2;
	}

	string subscriber_owner_code = 1;
	SubscriberType subscriber_type = 2;
	string serial_number = 3;
}


// Aanmelden
message Subscribe {
	message DisplayProperties {
		enum DestinationDetermination {
			MAX_CHARACTERS = 0;
			SELF_DETERMINING = 1;
		}

		uint32 text_characters = 1;
		bool overview_display = 2;
		DestinationDetermination destination_determination = 3;
	}

	message FilterParameters {
		bool filter_on = 1;
		uint32 waitingtime_low = 2;
		uint32 waitingtime_high = 3;
		uint32 percentage_low = 4;
		uint32 percentage_high = 5;
	}

	message FieldFilter {
		enum Delivery {
			NEVER = 0;
			ALWAYS = 1;
		}

		Delivery target_arrival_time = 1;
		Delivery target_departure_time = 2;
		Delivery expected_arrival_time = 3;
		Delivery expected_departure_time = 4;
		Delivery number_of_coaches = 5;
		Delivery trip_stop_status = 6;
		Delivery transport_type = 7;
```

```
                Delivery wheelchair_accessible = 8;
                Delivery is_timing_stop = 9;
                Delivery stop_code = 10;
                Delivery destinations = 11;
                Delivery show_cancelled_trip = 12;
                Delivery block_code = 13;
                Delivery occupancy = 14;
                Delivery line_public_number = 15;
                Delivery side_code = 16;
                Delivery line_direction = 17;
                Delivery line_color = 18;
                Delivery line_text_color = 19;
                Delivery line_icon = 20;
                Delivery destination_color = 21;
                Delivery destination_text_color = 22;
                Delivery destination_icon = 23;
                Delivery generated_timestamp = 24;
                Delivery journey_number = 25;
        }

        ClientId client_id = 1;
        repeated string stop_code = 2;
        DisplayProperties display_properties = 3;
        FilterParameters filter_parameters = 4;
        FieldFilter field_filter = 5;
        string description = 6;
}

message SubscriptionResponse {
        enum Status {
                REQUEST_INVALID = 0;
                STOP_INVALID = 1;
                AUTHORISATION_REQUIRED = 11;
                PLANNING_SENT = 20; // Successful
                NO_PLANNING = 21; // Successful, but no messages to send
                AUTHORISATION_VALIDATED = 22;
                DATA_CHANGED = 23;
                CONFIG_CHANGED  = 24;
        }
        bool success = 1;
        Status status = 2;
        uint32 timestamp = 3;
}

message Unsubscribe {
        ClientId client_id = 1;
        bool is_permanent = 2; // Last-will is not permanent
        uint32 timestamp = 3;
}
```

```
// Reisinformatie
message PassingTimes {
	enum TripStopStatus {
		PLANNED = 0;
		DRIVING = 1;
		CANCELLED = 2;
		ARRIVED = 3;
		PASSED = 4;
		UNKNOWN = 5;
	}

	enum TransportType {
		BUS = 0;
		TRAM = 1;
		METRO = 2;
		TRAIN = 4;
		BOAT = 5;
	}

	enum ShowCancelledTrip {
		TRUE = 0;
		FALSE = 1;
		MESSAGE = 2;
	}

	message Destination {
		repeated string destination_name = 1;
		repeated string destination_detail = 2;
	}

	repeated string pass_time_hash = 1;
	repeated uint32 target_arrival_time = 2;
	repeated uint32 target_departure_time = 3;
	repeated uint32 expected_arrival_time = 4;
	repeated uint32 expected_departure_time = 5;
	repeated uint32 number_of_coaches = 6;
	repeated TripStopStatus trip_stop_status = 7;
	repeated TransportType transport_type = 8;
	repeated bool wheelchair_accessible = 9;
	repeated bool is_timing_stop = 10;
	repeated string stop_code = 11;
	repeated Destination destinations = 12;
	repeated ShowCancelledTrip show_cancelled_trip = 13;
	repeated string block_code = 14;
	repeated uint32 occupancy= 15;
	repeated string line_public_number = 16;
	repeated string side_code = 17;
	repeated uint32 line_direction = 18;
```

```
        repeated string line_color = 19;
        repeated string line_text_color = 20;
        repeated string line_icon = 21;
        repeated string destination_color = 22;
        repeated string destination_text_color = 23;
        repeated string destination_icon = 24;
        repeated uint32 generated_timestamp = 25;
        repeated uint32 journey_number = 26;
}

message GeneralMessage {
        enum GeneralMessageType {
                GENERAL = 0;
                OVERRULE = 1;
                BLANC = 2;
        }

        enum ShowOverviewDisplay {
                TRUE = 0;
                FALSE = 1;
                ONLY = 2;
        }

        enum MessagePriority {
                CALAMITY = 0;
                PTPROCESS = 1;
                COMMERCIAL = 2;
                MISC = 3;
        }

        repeated string message_hash = 1;
        repeated GeneralMessageType generalmessage_type = 2;
        repeated string message_content = 3;
        repeated uint32 message_start_time = 4;
        repeated uint32 message_end_time = 5;
        repeated uint32 generated_timestamp = 6;
        repeated ShowOverviewDisplay show_overview_display = 7;
        repeated string message_title = 8;
        repeated MessagePriority message_priority = 9;
}

message GeneralMessageRemove
{
        repeated string message_hash = 1;
        repeated uint32 generated_timestamp = 2;
}

message PublicName
{
```

```
        repeated string stop_code = 1;
        repeated string public_name_place = 2;
        repeated string public_name_stop_place = 3;
        repeated string public_name_quay = 4;
}

message Container {
        PassingTimes passing_times = 1;
        GeneralMessage general_messages = 2;
        GeneralMessageRemove general_messages_remove = 3;
        PublicName public_names = 4;
}


// logboodschappen en meetwaarden

enum StatusType {
        ERROR = 0;
        WARNING = 1;
        OK = 2;
        LOG = 3;
}

message SystemStatus {
        message Metric {
                ClientId client_id =1;
                string component = 2;
                string component_index = 3;
                string property = 4;
                int32 value = 5;
                string unit = 6;
                uint32 timestamp_begin = 7;
                uint32 timestamp_end = 8;
        }

        message LogMessage {
                ClientId client_id =1;
                StatusType type = 2;
                uint32 code = 3;
                string message = 4;
                int32 duration = 5;
                uint32 timestamp = 6;
        }

        repeated Metric metrics = 1;
        repeated LogMessage logs = 2;
}
```

```
message InfoRequest {
        enum RequestType {
                SYSTEM_INFO = 0;
                STATUS = 1;
                TRAVEL_INFO = 2;
                SCREEN_CONTENT = 3;
        }

        ClientId client_id = 1;
        RequestType request_type = 2;
}

enum ValueType {
        STRING_TYPE = 0;
        INT_TYPE = 1;
        DOUBLE_TYPE = 2;
        BOOL_TYPE = 3;
        BYTES_TYPE = 4;
}

message SystemInfo {
        message SystemInfoLine {
              string name = 1;
                ValueType value_type = 2;
                string string_value = 3;
                uint32 int_value = 4;
                double double_value = 5;
                bool bool_type = 6;
                bytes bytes_value = 7;
        }
        ClientId client_id = 1;
        repeated SystemInfoLine system_info_lines = 2;
}

message StatusOverview {
        message StatusOverviewLine {
                string component = 1;
                string property = 2;
                ValueType value_type = 3;
                string string_value = 4;
                uint32 int_value = 5;
                double double_value = 6;
                bool bool_type = 7;
                bytes bytes_value = 8;
                StatusType status = 9;

        }
        ClientId client_id = 1;
        repeated StatusOverviewLine status_overview_lines = 2;
```

```
}


message TravelInfo {
        ClientId client_id = 1;
        Container travel_info_content = 2;
}


message ScreenContentResponse {
        enum ContentType {
                TEXT = 0;
                IMAGE_PNG = 1;
                IMAGE_GIF = 2;
        }
        message ScreenContent {
              uint32 screen_index = 1;
                ContentType content_type = 2;
                string content = 3;
                uint32 timestamp = 4;
        }
        ClientId client_id = 1;
        repeated ScreenContent screen_contents = 2;
}

message FileAvailable {
        ClientId client_id = 1;
        string file_name = 2;
}
```