Recommendation 453

3/08

Real Time Interface
VDV 453

Former titele: Integration Interface for Automatic Vehicle Management Systems

Version 2.3e

- → **Connection protection**
- → **Dynamic passenger information**
- → **Visualisation**
- → **General message service**

Verband Deutscher Verkehrsunternehmen (VDV)

Kamekestraße 37 - 39, 50672 Köln, Tel. 0221/57979-0, Fax: 514272

Client

Federal Ministry for Traffic, Construction and Housing
Invalidenstrasse 44
D -10115 Berlin

Contractor

Beratungsgesellschaft für Leit-, Informations- +Computertechnik mbH (BLIC)
Rheinstraße 45
D - 12161 Berlin

Mr. Gustav Thiesing (Dipl.-Ing)

Die Änderungen gegenüber der Vorversion wurden beraten und beschlosen in der **VDV Arbeitsgruppe "Ist-Daten-Schnittstellen"** u. a. von:

Bayer, Marcus	Deutsche Bahn AG	München
Beck, Michael	initplan GmbH	Karlsruhe
Bibergeil, Wolfgang	Funkwerk Information	Kiel
Braun, Volker	IVU Traffic Technologies AG	Aachen
Eckardt,	Eckardt Software Management GmbH	Hannover
Elsensohn, Peter	Technische Informationssysteme GmbH	Rankweil
Fiekert, Wolfram	HaCon Ingenieurgesellschaft mbH	Hannover
Frankenberg, Michael	HaCon Ingenieurgesellschaft mbH	Hannover
Jürgens, Sven	PSI Transcom GmbH	Berlin
Kohl, Werner	Mentz Datenverarbeitung GmbH	München
Lenzen, Karl Horst	T-Systems GEI GmbH	Mülheim
Lisbach, Bettina	init GmbH	Karlsruhe
Martinez-Dreyer, Günther	PSI Transcom GmbH	Berlin
Müller, Achim	STAM	Buxtehude
Pirker, Oswald	DB Systel GmbH	Frankfurt
Rubli, Daniel	Continental Automotive	Neuhausen am Rheinfall
Tödt-Nissen, Karl-Heinz	HanseCom	Hamburg
van der Worp, Bart	intraffic	MR Nieuwegein
van Sorgen, Atze	MRK Management Consultants GmbH	München
Weik, Friedemann	Hamburger Berater Team GmbH	Hamburg

Change history:

In this version the translation of the german tag AbbringerFahrtLoeschen has been changed from CPITripDelete to FetcherTripDelete.

1 Foreword

This is a translation of the VDV recommendation 453. It is intended to help non german readers to apply the specification. In the case of differences between this translation, the german original document and the xsd schema: The schema (available on www.vdv.de) is the binding document!

VDV 453 has been drafted within the context of the urban traffic research project (FOPS) 70.0701/2002 of the Federal Ministry for Traffic, Construction and Housing (BMVBW) by BLIC and IAV under the technical supervision of the VDV. Partners from the AVMS industry and various transport authorities were also involved with the specification. This paper, "Integration Interface for Automatic Vehicle Management Systems" sets down common requirements for the design of interfaces between automatic vehicle management systems (AVMS). In addition to the main and cross-functional requirements, there is also a detailed description of the requirements on data exchange.

The general specification aims to promote uniformity, modularity and manageability of the software and hardware for the purpose of this comprehensive data exchange.

The motivation for an extension of version 1.0 of March 2001 to version 2.0 arose from experiences gained during implementation. Version 2.0 of the interface has been technically revised, restructured and supplemented with additional specialist services. A common technical communications infrastructure has been created for all technical services, which ensures implementation and extension are less time consuming and more cost effective than in version 1.0. The new architecture is open for further services.

The following services have been specified:

Service	**Purpose**	**Comments**
Reference data service for connection protection (REF-CP)	Exchange of planned schedules for connection protection	Already included in VDV 453 version 1.0, revised
Process data service for connection protection (CP)	Exchange of real-time data for connection protection	Already included in VDV 453 version 1.0, revised
Reference data service for passenger information (REF-DPI)	Exchange of location related planned schedules for passenger information	New to version 2.0
Process data service for passenger information (DPI)	Exchange of real-time data for passenger information	Already included in VDV 453 version 1.0, revised
Process data service for visualisation (VIS)	Exchange of real-time data for the visualisation of vehicles in foreign control centres	New to version 2.0
General message service (GMS)	Exchange of textual information between the control centres	New to version 2.0

In this Version 2.3 mistakes of the previous versionhave been mended and some functional enhancements added..

Reference for the implementation of services is the latest XML-Schema (view VDV-Internetsite www.VDV.de), which contains all the services of this document and also the services of VDV454, Schedule Information: REF-SIS service and the SIS Real-time Data Service. This document has to be viewed as explanation only. The time being, it exist only a german version of the XML-Schema. For this reason the definitions in this document have the german xml-tag in brackets, in order to see the connection between this translated document and the xsd file.

2 Background

2.1 The Project

The aim of the "Integration Interface for Automatic Vehicle Management Systems" project is the implementation of an interface to link AVMS systems from various transport authorities and possibly also different manufacturers. Many transport authorities working in urban and regional areas have installed AVMS systems in recent years to optimise and improve their operational processes. To a certain extent, the manufacturers of these systems follow different strategies with regard to the associated hardware and software, which creates very different system designs. The problem is that the functional boundaries and system resources of the individual AVMS systems are not prepared for matching processes with other transport authorities. There are system boundaries between the individual AVMS, but these boundaries should not be obvious to the passenger, who demands consistent quality standards and a functioning unified transport system. To be able to implement connection protection between neighbouring transport authorities for example, it is necessary to coordinate the operational processes at the corresponding interfaces of the individual systems. System variety and individual design mean there is a real danger of an unacceptable amount of work for each separate system coupling. The aim of this paper therefore is to outline a common mechanism with a common interface for the integration of AVMS systems, which will allow the individual operators to implement such functionality at a reasonable cost, in terms of both procurement and operation.

The technical side of the concept is based on standard technologies (HTTP/XML). These are responsible for the essential tasks of an interface. They firstly reduce the costs of implementation and also ensure the necessary compatibility where several different manufacturers are involved.

Transferability – even to small AVMS systems – is guaranteed, as both the technologies used and the amount of data to be exchanged require few resources.

The essential "softer" requirements on the inter-operational organisation, which would be introduced by such a concept, include establishing

→ Inter-operational income,

→ Inter-operational financial coordination

→ A matched inter-operational (partial) database

2.2 Objectives

When dealing with control technology systems, the operator is more often than not faced with the need to integrate into existing technical control configurations. At the same time however, the system concept must be designed so that it takes into consideration the latest technological status. From the point of view of the further development of the control technologies, it is also necessary to consider that the automation of operational management is an unavoidable prerequisite with regard to the economic and efficient operation of local public transport networks. Transport authorities are increasingly following a policy of fully automated operational management, including the directly associated procedures in the context of preparation, implementation and follow-up work. The aim is to avoid as far as possible the need for intervention by operating staff. The result is an increasing interlinking of the organisational and technical systems.

A further important aim in connection with the introduction of new technologies is the improvement of the economic efficiency of procurement, operation and maintenance.

2.3 Model

The technical control equipment in transport authorities is usually distinguished by a long life, as a result of the levels of complexity and investment. Step-wise renovation and subsequent extension have led to the inevitable situation in which different generations of technical development can be found working alongside one another within many operations. Particularly where the system is not to be decommissioned, there is a need for continued operation and often compatibility with new systems.

For this reason, it must be possible to integrate any new control equipment into the existing configuration of technical systems and it must support the operational procedures already in place. The principle of “least possible cost with the greatest possible benefits” is also significant, as it represents the foundation for effectiveness and profitability and at the same time reflects the entrepreneurial objectives of many transport authorities.

In view of progressing automation, the model must also correspond to the known requirements. This situation demands that the technical solution have a modular design and that it offers the possibility of step-by-step implementation of the given objectives.

3 Introduction and Basic Terms

3.1 Connection Protection (CP)

3.1.1 Tasks and Objectives

Direct management of all desired trip relationships is practically impossible, particularly in major cities and urban conglomerates, and from an economic point of view unfeasible. When planning the network, the services and the individual schedules, a transport authority must take into consideration, in accordance with the predominant traffic flow, the connection relationships between the individual lines in the planned schedules in the form of connection planning.

The basic problem when planning connections is that for the passengers in “broken journeys”, i.e. those journeys associated with a need to transfer from one vehicle to another, the changeover times represent a significant comfort or acceptance factor. There is a need to guarantee the least possible waiting time for each individual trip relationship. If the waiting times are too long, there is a danger that the passenger will not accept the planned connection. When defining connections, it is also necessary to take into consideration the time and location related limitations (journey times between the arrival stop and the new departure stop) as well as the operational and other time-of-day specific considerations that apply to the majority of all connections.

This problem, which is largely manageable for two lines and one interchange point, leads to the fact that for many connection points and associated lines, it is only possible to include a selection of interchanges and trip relations as planned connections within the schedules.

A connection always requires two vehicles – the so-called “feeder” and “fetcher”.

3.1.2 The Feeder-Fetcher Principle

Internal connection protection within an individual operation is already an established dispatch function. By comparing the schedule statues at a fixed changeover time and a particular connection stop (as required by the passengers), the fetcher vehicle can be instructed to wait for the feeder vehicle for a defined period of time.

The solution for implementing connection protection is based on the feeder-fetcher principle. There are always two vehicles involved in a connection relationship. With the mutual referencing of feeder-fetcher relationships (i.e. a feeder is also simultaneously a fetcher), it is also possible to realise multi and block connections.

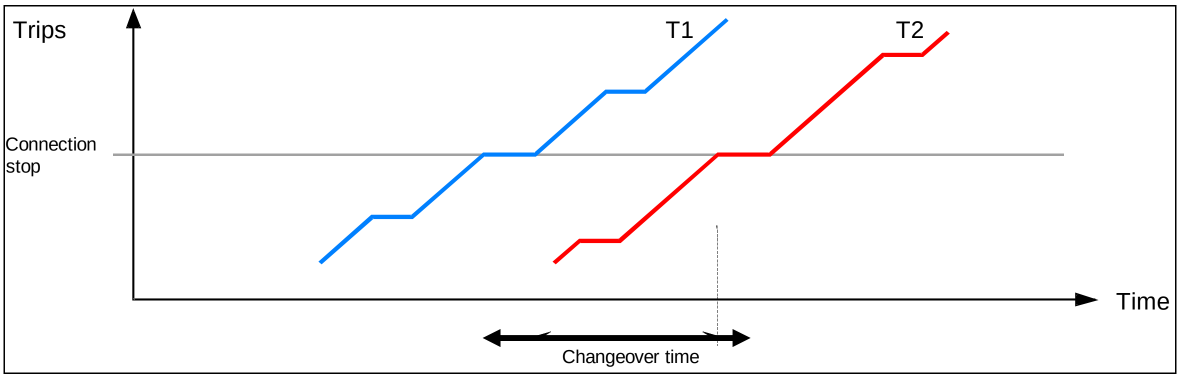

Figure 1: Feeder – fetcher connection

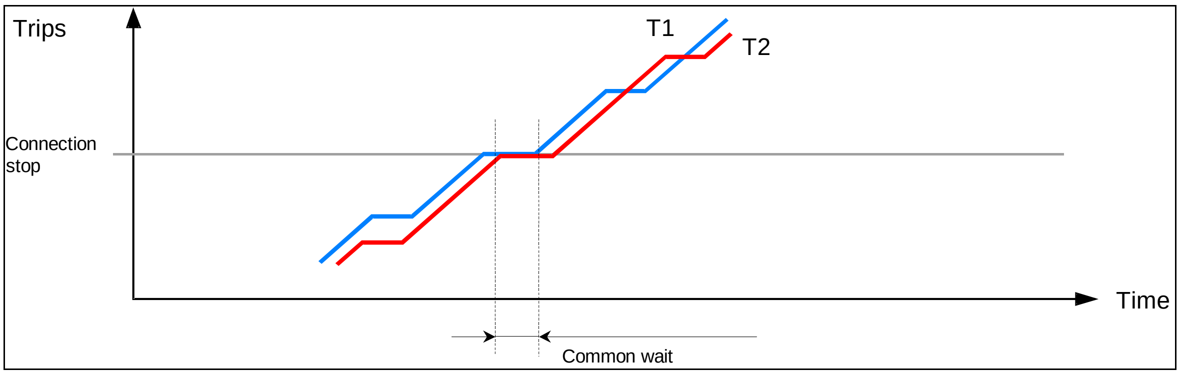

Figure 2: Block connection

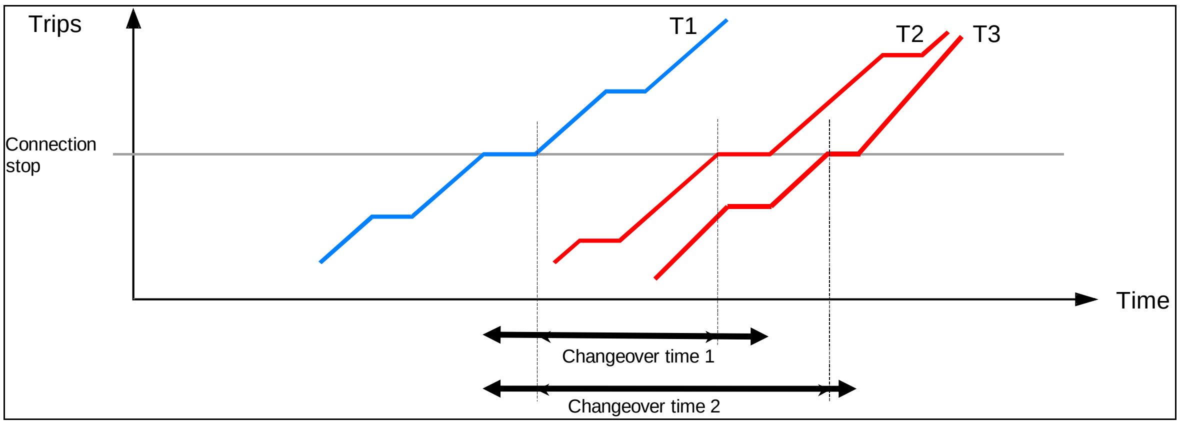

Figure 3: Multi-connection

3.1.3 Definition of Inter-Operational Connection Protection

As with internal connection protection, inter-operational connection protection should be implemented as automatically as possible by means of data exchange between the dispatch systems of the participant transport authorities where the dispatchers are only required to intervene in case of doubt (violation of a limiting condition). The basic prerequisite for automatic connection protection is that the AVMS that is to exert an influence on the fetcher vehicle (e.g. send it an instruction to wait for a delayed feeder vehicle) must have the necessary dispatch aid and functions within its software. The feeder AVMS does not require this functionality unless it is to assume fetcher functionality in the reverse situation.

The coordination process between the transport authorities must include a uniform, bilateral definition of the connections.

3.1.4 Operational Models

3.1.4.1 Stations

Stations are usually composed of several platforms. Defining the station as a single stop has the disadvantage that it is not possible to specify separate changeover times from a platform to a bus stop within the station complex. It is better to use modelling which defines the individual platforms as stops. This however is associated with the problem that trains are sometimes diverted from their regular arrival platform for operational reasons, and with that fail to reach the designated connection area. This demands corresponding computation in the feeder system that ensures any platform change appears transparent to the fetcher. The feeder system continues to report predictions concerning the originally planned connection area to the fetcher, even though the feeder has not actually reached it. Internally, this is achieved by a means of mapping the "old" feeder stop/platform onto the “new” stop/platform.

3.1.4.2 Multi-connections

Many AVMS systems permit the definition of so-called block or system connections (n:m relationships). This involves several trips at several stops with different feeder and fetcher functionalities.

With trip-related connection protection (see chapter 3.1.8), these connections must be split into individual connections (1:1 relations). Each fetcher is assigned to a single feeder.

With time-related connection protection (see chapter 3.1.9) a fetcher can have several feeders (n:1 relationship).

3.1.4.3 Multi-Serviced Stops

If a stop is called at more than once within a trip, location and trip code are no longer sufficient to uniquely identify an arrival. To combat this there is an additional counter. The principle is that arrival at a stop has a lesser counter value than a subsequent arrival at the same stop. The counter need not increase sequentially. It is possible to use the connection time (coded as an integer) or the stop index.

3.1.5 Schedule and Connection Planning (Planned Schedules)

This paper describes the technical interface of connection protection. This technical implementation replaces the need for any internal matching or inter-operational coordination between different transport authorities. For the planning and implementation of connection protection, all operational and traffic related limiters must be established in advance.

Using the created schedules, it is possible to identify and define possible connection points as inter-operational or common specifications.

It is necessary to establish the participant lines and their functions (feeder or fetcher).

It is then necessary to name the time range (e.g. via times or trip codes) in which the connection is to be protected. The background to this is a meaningful restriction – from the point of view of the passenger – to important connection times, particularly late at night or at times with long headways. In this connection it is necessary to consider what happens when a connection vehicle is held back and whether this endangers subsequent connections.

Depending on the local conditions, it is also necessary to specify the changeover times required by the passengers to transfer from one stop to another.

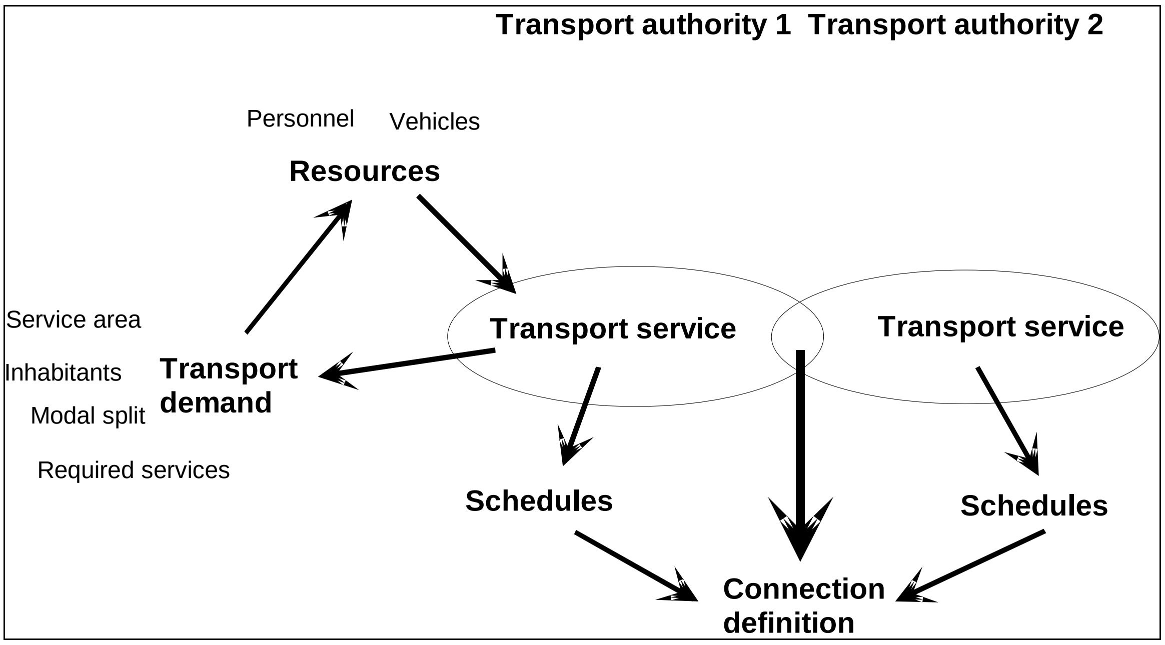

Figure 4: Schedules and connection definitions

3.1.6 Connection Areas

Connection areas are used to define connection relationships. They replace the direct modelling of the internal stop names of the respective systems. This effectively decouples the system databases. At the same time, there is an abstraction of the meaning of the internal key. In this way, it is possible to directly combine entire stop areas of two operations into one connection relationship.

In the simplest case, a connection area consists of two stops, a feeder stop and a fetcher stop. By definition, the changeover time is then zero.

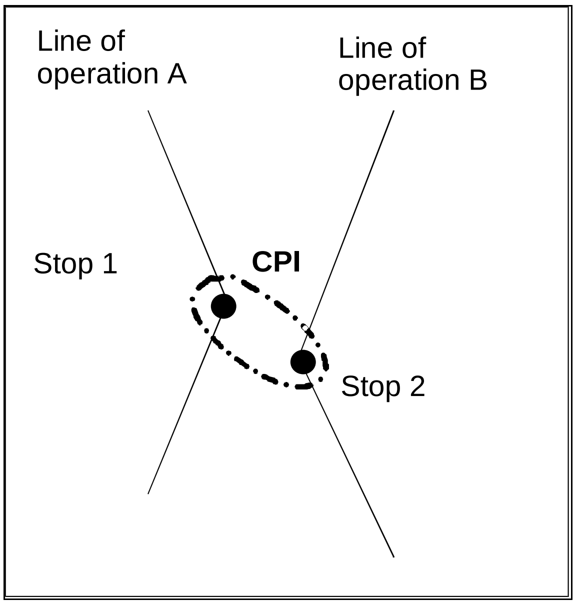

Figure 5: Connection area (CPI)

If changeover times are to be defined, then on the fetcher side, the depiction of the actual stop in the connection area must be supplemented with the changeover time. The fetcher system must add the changeover time to the predicted arrival time to establish the actual time at which the passengers arrive at the fetcher stop. Likewise, it is possible to assign several fetcher stops with different changeover times to one connection area. In terms of operational data management, changeover times represent attributes when allocating stops to connection areas. Changeover times are only defined on the fetcher side.

The structural situation often permits the allocation of several feeder and/or fetcher stops to one connection area. This saves effort and resources in operational data management. The rule is that all feeder stops that are to be assigned to a connection area must have the same changeover times to a fetcher stop that is also assigned to the connection area. This must hold true for all fetcher stops.

The following table shows the changeover times for a possible connection situation (with 3 feeder stops and 3 fetcher stops). As the columns Fdr1 and Fdr2 are identical, they can both be assigned to the same connection area. Any fields not occupied (no changeover) represent placeholders for any value, i.e. they act as “jokers”.

Stops	**Fdr1**	**Fdr2**	**Fdr3**
Ftr1	1 min.	1 min.	1 min.
Ftr2	2 min.	2 min.	4 min.
Ftr3	1 min.	No changeover	7 min.

Table 1: Matrix of changeover times for connection area definitions

Instead of the nine different connection areas in the given example (number of feeder stops multiplied by the number of fetcher stops) there are only two:

CPI1: Feeders Fdr1/Fdr2, fetchers: Ftr1/Ftr2/Ftr3

CPI2: Feeder Fdr3, fetchers: Ftr1/Ftr2/Ftr3

The data management now establishes the stops between which connections are actually required and the criteria these connection relationships depend on. Connection relationships are generally defined line-wise and specific to direction (e.g. from line 10, direction X to line 1, direction Y). In addition, connection relationships often only need to be protected at certain times of the day. The allocation tables for the connection areas must be supplemented accordingly.

3.1.7 Passenger Information on Interior Displays

When approaching an interchange point during a trip, the relevant information concerning the occurrence or failure of planned connection relationships should be collected in the vehicle. This information should also be retained, so that it can be provided in response to an enquiry. It should be announced both visually and audibly at suitable points before the connection area.

3.1.8 Trip-Related Connection Protection

Trip related connection protection corresponds to the approach taken in VDV 453 version 1.0. The planned schedules (reference data) are exchanged at the start of each operational day. In the subsequent phase of process data exchange, only the current delays are communicated.

Since VDV 453 version 2.0 updates to the planned data are also possible. This means any additional trips can be communicated at short notice.

The incorporation of severely delayed trips (replacement for follow-on trip) remains forbidden.

The advantage of this method is the early definition of the connections. This allows the passengers to be informed of connections taking place in advance.

3.1.9 Time-Related Connection Protection

In comparison with version 1.0, time-related connection protection represents a totally new approach. This allows connections to be protected without any prior exchange of reference data. Here the request to the feeder system is made during the actual approach of a fetcher vehicle to the connection area. The request defines a place and time range, which restricts the number of trips on the basis of their predicted arrival time at the given connection area. Line and direction filters can also be used to restrict the volume of data.

Time-based connection protection can therefore cover all dispatch actions that lead to a change to the arrivals display board at the feeder stop. This means that even very delayed vehicles can become feeders.

A possible disadvantage is the later awareness of the connection definition. However, with suitable definitions in the data management, it is possible to achieve equal planability in the definition of connections.

3.2 Dynamic Passenger Information (DPI)

3.2.1 Tasks and Objectives

The aim of the "Dynamic Passenger Information (DPI)" service is to create the possibility of displaying foreign trips at commonly served stops. Data exchange is location specific, i.e. it involves the transfer of departure tables for locations that have been commonly defined in advance.

A differentiation is made between passenger information in the vehicle and at the stop. The connection information in feeder vehicles is implemented within the context of the connection protection service. The visual and dynamic passenger information at stops has become an integral element of automatic vehicle management systems.

The DPI service is described with the following elements:

→ Individual predictions for each display (location specific predictions)

→ Quick or regular cleardown (radio or AVMS message)

→ Start / destination / via texts (codes and full texts)

→ Special trip-related texts

→ Special line-related texts

→ Text length specification

→ Preview times for meaningful communication

3.2.2 Data Supply and Control

The DPI service addresses the common use of DPI displays. A third-party operator wishes to display the arrival/departure of a vehicle on the displays of a different transport authority. The system that controls the displays is called the “*display owner*” and the system that wishes to represent the vehicle on a foreign display is called the “*display user*”.

Two different procedures are covered by the technical DPI service:

→ Controlling individual displays with full text information

→ Controlling individual displays with code values instead of full texts

The full text method is the simplest way to supply displays with data. It is best suited to situations with few displays to be controlled and where this is permitted by the available bandwidth between the AVMS and displays.

The variant involving coded supply can be used when the bandwidth between the AVMS and displays is critical. The use of codes significantly reduces the bandwidth requirements.

In its simplest form, passenger information does not require the exchange of reference data. Data exchange is only required at run time, i.e. in live operation. However, this does demand a list of connection areas, which allows the AVMS to determine where the “foreign” displays are provided, upon which information is to be displayed. Some systems do also require access to planning data in order to be able to process foreign trips during operation. For these systems there is an additional reference data interface, which makes the required schedule information available.

Fundamentally speaking, the "display owner" is responsible for determining which information is displayed. The foreign AVMS supplies the relevant display area information, which contains the following details:

→ Trip ID

→ Arrival time

→ Departure time (optional)

→ Line and direction text

→ Stop position (e.g. platform 5, bay C, etc.)

→ Trip destination

→ Via details

→ Comments

→ Special line texts

→ Special trip texts

The controlling of display systems on the basis of codes demands a common operational data supply. (Foreign) texts and associated codes must be made known to the display owner system. As before, full texts are transferred via the interface. Thanks to the availability of tables on the display owner side, the corresponding codes can be established by comparing the texts and then sent to the displays.

3.2.3 Display Areas

In a similar way to connection protection, the common usage of displays requires a common referencing system. For each commonly used display (or group of displays) a so-called *display area* is defined with a unique code (*DISID*).

The display user system links its own stops located within the display area with the display area code, and with that can connect arrivals or departures at a stop with a foreign display. The display owner system links the internal codes of its own displays with those of the commonly defined display areas. This closes the reference chain between the display user and display owner.

3.3 Visualisation of Foreign Vehicles (VIS)

3.3.1 Tasks and Objectives

The “Visualisation of Foreign Vehicles” (VIS) service supports control centre employees in the monitoring and control of foreign vehicles within their own AVMS system. This service offers the possibility of transferring information about these foreign vehicles so that they can be represented in the tables and diagrams of their own system. It provides the following data:

→ Geographical position of the vehicle (map representation)

→ Position of the vehicle in relation to its route path (line diagram)

→ Display in a vehicle list

→ Further details concerning the trip (start and destination stops, service characteristics, etc.)

Schematic line diagrams require a background route path model of the trip. This can be achieved by supplying dummy routes within the proprietary system. This interface does *not* provide direct dispatch functions. Other channels must be used (radio). In addition, the line diagram demands a common route path supply. The interface does not cover this functionality either, and it must be achieved in a different way, if such a method of representation is required.

3.3.2 Visualisation Areas

As the representation of foreign vehicles is usually selective (e.g. related to line groups), the interface offers the possibility of also requesting the data selectively. For this, so-called *visualisation areas* are formed. These are abstract designators that define the type and amount of data to be requested.

Visualisation areas are usually defined specific to line. A visualisation area is assigned one or several lines. In response to a request, data concerning the lines associated with the visualisation area is exchanged. Alternatively, a visualisation area can also be defined spatially (by area) or in a trip-specific way. This may be associated with a lengthy real-time check, which assigns vehicles to the visualisation areas.

Visualisation areas must be jointly defined by all participant systems in accordance with the operational specifications.

3.4 General Message Service (GMS)

3.4.1 Tasks and Objectives

The "General Message Service (GMS)" is used by the employees of the participant control centres to exchange general operational information. It provides a means of exchanging text messages and therefore represents an alternative to the fax or email. The advantages with regard to the more conventional methods include immediate transmission and simple integration into the control centre software.

Messages can be sent and also revoked (deleted at the recipient location). Messages are assigned validity in addition to the actual content.

3.4.2 Message Channels

Within the message service, every control centre message is assigned to a so-called message channel. The message channel represents a bilateral agreement with regard to the classification of the content.

This allows the messages to be displayed and managed in accordance with the requirements of the control centre employees.

3.4.3 Message Formats

The message service can be used to transmit structured (CSV, XML, etc.) and unstructured information. The XML content model permits a free design.

There is also a format attribute that permits the automatic detection of various different formats.

4 Architecture

4.1 Communication vs. Technical Services

The interface consists of two layers:

1. Communication layer
2. Technical service layer

The communication layer defines a common procedure for requesting and then exchanging data. This procedure is hereafter referred to as the *subscription process*. The data consumer system creates so-called subscriptions, which define the type and volume of data to be exchanged. This definition is technically specific and therefore needs to be established in the technical layer. The communication procedure is the same for all services and with that represents the interface infrastructure (message referencing, error handling, reset behaviour). Reusing it for the various technical services ensures cost-effective implementation and extension of the interface.

The technical services are in turn built on the communication layer and address various application areas such as connection protection, dynamic passenger information, etc. The services are independent of one another, allowing any number of services to be implemented. This guarantees application-specific implementation.

4.2 Reference Data vs. Process Data

The data exchanged via the technical services can be split into two classes:

→ Reference data (planning data)

→ Process data (real-time data)

Every technical service (connection protection, dynamic passenger information, etc.) must always facilitate real-time data exchange, as they deal with the exchange of current information between two control systems. The exchange of reference data is only necessary when the process data cannot be used on its own, but where it needs to be related to reference data first.

The exchange of reference data generally represents an alternative to the exchange of data at the level of data management. The implementation and use of a reference data service depends on the application case as well as the technical operational requirements. Each individual service can therefore consist of two separate technical services. Within the interface, the process data exchange and reference data exchange are implemented as two separate technical services.

4.3 Protocols

Two protocols are used in the interface:

1. HTTP/1.1 as the transport protocol
2. XML 1.0 for recording the technical data

The XML specification is based on the XML schema version 1.0.

.

5 Interface Description of the "Basic Infrastructure"

5.1 Subscription Procedure

5.1.1 Summary

The so-called subscription method defines a common basic communication structure, on which all technical services are based. The subscription procedure consists of a set of request and reply messages, which define an asynchronous communication structure.

The concept follows the client-server model. System A (server) can make data available to another system B (client).

The concept is event-based. Data changes as a result of an action in the server system (A), which then needs to be communicated to the client system (B) (see Figure 1).

The client and server first agree which information is to be exchanged. This is achieved by so-called subscriptions. Subscriptions are defined on the client side. The client sends a subscription request to the server and with that registers interest for specific data (step 1). The data concerned is defined within the actual subscription request. After confirmation from the server, the client can expect a subsequent supply of data.

The server (A) then informs the client (B) about new or modified data by means of a corresponding message (step 2). The client (B) can then retrieve the corresponding data from the server (A) (step 3).

In order to detect a server breakdown, status requests can be periodically sent to the server. With a status reply, the server confirms its functionality (step 4).

Subscriptions have a life span as defined by the client and once expired are automatically deleted by the server. Deletion can also be achieved prior to this by the client (step 5).

The services are managed separately, according to service type. Subscriptions are referenced via so-called *SubscriptionIDs*. A *SubscriptionID* is unique within a service. The client is responsible for assigning *SubscriptionIDs*.

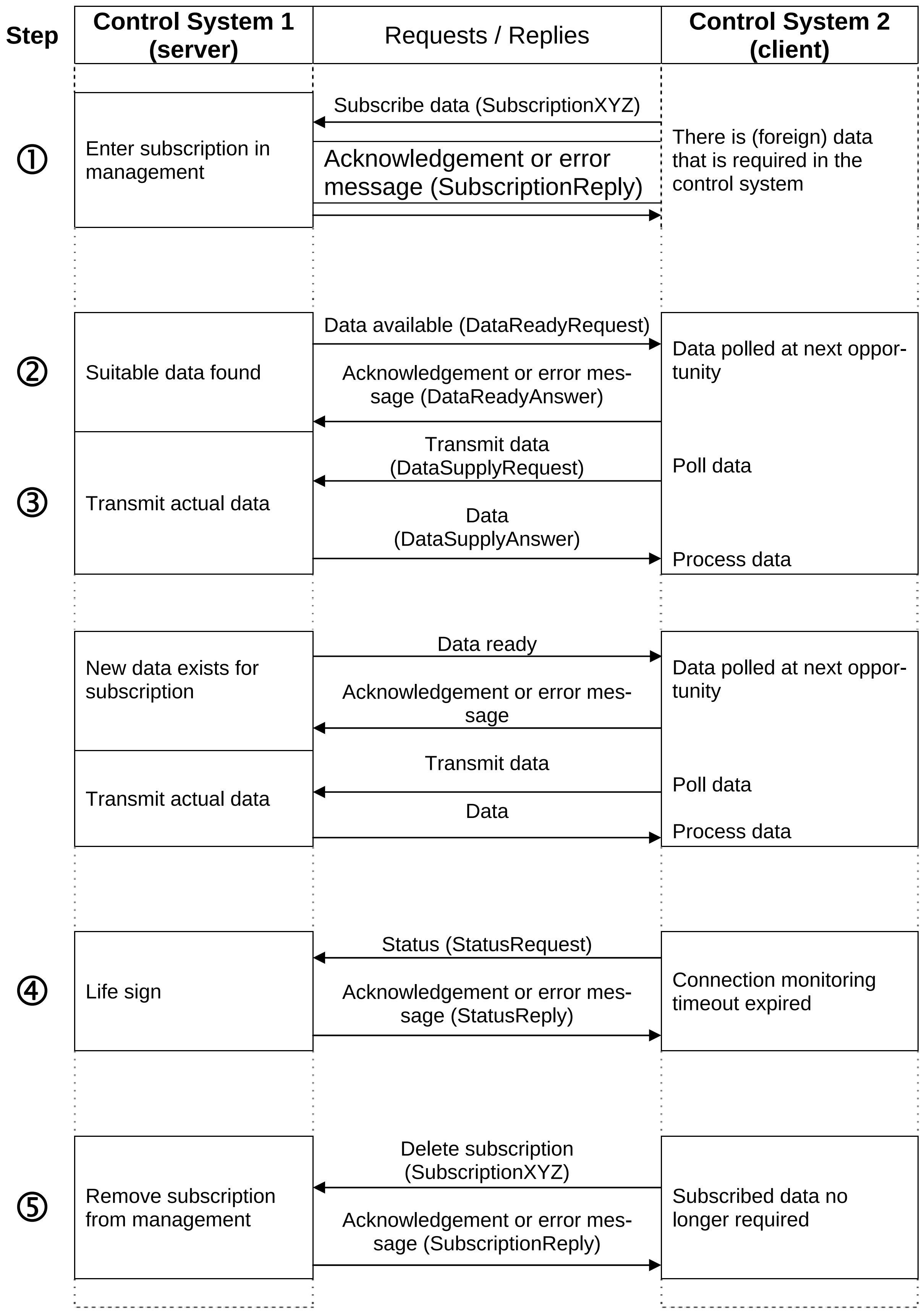
Step
Control System 1 (server)
Requests / Replies
Control System 2 (client)
①
Enter subscription in management
Subscribe data (SubscriptionXYZ)
Acknowledgement or error message (SubscriptionReply)
There is (foreign) data that is required in the control system
②
Suitable data found
Data available (DataReadyRequest)
Acknowledgement or error message (DataReadyAnswer)
Data polled at next opportunity
③
Transmit actual data
Transmit data (DataSupplyRequest)
Data (DataSupplyAnswer)
Poll data
Process data
New data exists for subscription
Data ready
Acknowledgement or error message
Data polled at next opportunity
Transmit actual data
Transmit data
Data
Poll data
Process data
④
Life sign
Status (StatusRequest)
Acknowledgement or error message (StatusReply)
Connection monitoring timeout expired
⑤
Remove subscription from management
Delete subscription (SubscriptionXYZ)
Acknowledgement or error message (SubscriptionReply)
Subscribed data no longer required

Figure 1: Communication procedure for subscriptions

5.1.2 Setting Up Subscriptions

Subscriptions are to be created wherever data needs to be retrieved from an external system. This is achieved by the client sending a so-called *SubscriptionRequest*.

5.1.2.1 SubscriptionRequest

A *SubscriptionRequest* defines one or more subscriptions (sub-elements). Every subscription has an identification that is unique for the service (*SubscriptionID*). This is generated and managed by the client. Because several subscriptions can be set up within one SubscriptionRequest, but there is only one general error message for the entire SubscriptionRequest process, the procedure is as follows when an error occurs:

If it proves impossible to set up at least one subscription of a SubscriptionRequest, a corresponding error message is returned. The error text should contain an exact description stating which subscriptions could not be created.

If an error occurs within a SubscriptionRequest, this renders the entire request invalid and it is rejected. It is not possible to set up or delete subscriptions.

Implementation hint: To achieve a data exchange during implementation or first connection tests, it may be senseful to put each subscription in a separate SubscriptionRequest. Thus mistakes can be recognized more easily.

In the case of a successful SubscriptionRequest, it is also necessary to set up all subscriptions on the sender side too.

Within the subscription, service specific sub-elements define the possible replies. All messages to define a subscription follow naming convention SubscriptionXYZ, where “XYZ” stands for the corresponding service code (CPI, DIS, VIS), or the corresponding action.

The following subscription requests are defined:

- → Connection protection: Subscription of feeder data (*CPISubscription 6.2.4.1.2, CPIRefSubscription 6.2.3.2*)
- → Passenger information: Subscription of data for displaying approaching trips of foreign operators on the DPI displays (*DISSubscription, 6.3.8.2*).
- → Visualisation: Subscription of trip data for the visualisation areas (*VISSubscription 6.4.3.2*).
- → Deletion of one or all subscriptions (*DeleteSubscription / DeleteSubscriptionsAll 5.1.5*).
- → Message service: Subscription of general text messages from the foreign control centre (GMSSubscription 6.5.4.1).

Definition of *SubscriptionRequest (AboAnfrage)*:

Sender (Sender):	(attribute) Control centre code of the requesting system
TimeStamp (Zst):	(attribute) Time stamp at creation of request
CPIRefSubscription (AboASBRef):	(sub-element) Subscribes reference data for connection protection
CPISubscription (AboASB):	(sub-element) Subscribes process data for connection protection
DISSubscription (AboAZB):	(sub-element) Subscribes process data for passenger information
DISRefSubscription (AboAZBRef):	(sub-element) Subscribes reference data for passenger information
VISSubscription (AboVIS):	(sub-element) Subscribes process data for visualisation
DeleteSubscription (AboLoeschen):	(sub-element) Deletes a single subscription
DeleteSubscriptionsAll (AboLoeschenAlle):	Deletes all subscriptions
GMSSubscription (AboAND):	(sub-element) Subscribes process data of the message service

A *SubscriptionRequest* can only contain sub-elements *of one* specific service. If a SubscriptionRequest is created with a *SubscriptionID* that already exists, the existing subscription is overwritten.

All subscriptions of all services are given a time stamp (ValidUntilTimeStamp) by the client when set up. This defines how long the server must save and manage the subscriptions. The time stamp should be selected so that it lies behind the last potential data registration time point. Subscriptions should not be given a longer validity than is necessary as each subscription makes demands on the resources. Subscriptions can have validities that cross the boundaries of the given operational day.

The following example shows the request from AVMS A to set up two subscriptions with IDs 25 and 26 with a validity of one hour.

```
<SubscriptionRequest Sender="AVMS A" TimeStamp="2001-08-08T05:00:00">
  <SubscriptionXYZ SubscriptionID="25" ValidUntilTimeStamp="2001-08-
08T06:00:00">
    <Info1> ... </Info1>
    <Info2> ... </Info2>
  </SubscriptionXYZ>
  <SubscriptionXYZ SubscriptionID="26" ValidUntilTimeStamp="2001-08-
08T06:00:00">
    <Info1> ... </Info1>
    <Info2> ... </Info2>
  </SubscriptionXYZ>
</SubscriptionRequest>
```

5.1.2.2 SubscriptionReply

After the data producer (server) has received the request, it acknowledges with a *SubscriptionReply* message.

Definition of *SubscriptionReply (AboAntwort):*

Acknowledge (Bestaetigung):	(sub-element) Contains information for error handling

The following example shows the successful confirmation of the above request:

```
<SubscriptionReply>
      <Acknowledge>
            TimeStamp="2001-08-08T05:00:10"
            Result="ok"
            ErrorNumber="0">
      </Acknowledge>
</SubscriptionReply>
```

The *Acknowledge* element supplies information on whether the request could be processed, any possible error codes and descriptions, information on the maximum possible update rate of the data producing system as well as the available data horizon (6.1.1).

Definition of *Acknowledge (Bestaetigung)*:

TimeStamp (Zst):	(attribute) Time stamp defining creation of the acknowledgement
Result (Ergebnis):	(attribute) « ok » without error, « notok » if the request could not be processed
ErrorNumber (Fehlernummer):	(attribute) Number for a more exact classification of the error (see 6.1.10).
Errortext (Fehlertext):	(optional) Description of the error in plain text
DataValidUntil (DatenGueltigBis):	(optional) End of the data horizon of the data producer; omitted if the request lies completely within the data horizon.
ShortestPossibleCycleTime: *(KuerzMoeglicherZyklus)*	(optional) Minimum separation between two updates depends on the processing cycle of the AVMS

When the data consumer (client) receives the *Acknowledge* it knows the subscription has been set up. Otherwise the request must be re-transmitted. This overwrites any subscriptions which may have been set up on the server side.

5.1.3 Signalling Data Availability

5.1.3.1 DataReadyRequest

Once the subscription has been set up and the data made available, the data consumer is informed of the existence of updated data via a *DataReadyRequest*. This occurs with every change to data associated with the subscription. The signalling relates to all subscriptions of a service.

Definition of *DataReadyRequest (DatenBereitAnfrage)*:

Sender (Sender):	(attribute) Control centre code of the data producing system
TimeStamp (Zst):	(attribute) Time stamp of the notice of change message

The following example illustrates the signalling of new data in a service of AVMS A:

```
<DataReadyRequest
      Sender="AVMS A"
      TimeStamp="2001-08-08T08:00:00">
</DataReadyRequest>
```

5.1.3.2 DataReadyAnswer

The data consumer (client) confirms reception of signalling with a DataReadyAnswer type message. This message contains an Acknowledge element:

Definition of *DataReadyAnswer (DatenBereitAntwort)*:

Acknowledge (Bestaetigung):	(sub-element) Contains information for error handling

The following example shows a possible reply to the above request:

```
<DataReadyAnswer>
      <Acknowledge
            TimeStamp="2001-08-08T08:00:10"
            Result="ok"
            ErrorNumber="0">
      </Acknowledge>
</DataReadyAnswer>
```

The data can now be retrieved by the client. If the client does not wish to retrieve data at this particular moment, it can be postponed to a later time. Data polling occurs independently of the signalling of changed data.

5.1.4 Polling Data

Data polling occurs at the initiative of the data consumer (client). Only the current real-time information is transmitted. Historical data is not available.

5.1.4.1 DataSupplyRequest

Polling generally occurs after updated data is signalled (*DataReadyRequest*), but can occur at any time after setting up the subscription. The client sends a *DataSupplyRequest* message, which prompts the server to supply the data that has been updated since the last *DataSupplyRequest*:

Definition of *DataSupplyRequest (DatenAbrufenAnfrage)*:

Sender (Sender):	(attribute) Control centre code of the data producing system
TimeStamp (Zst):	(attribute) Time stamp of the request
AllData (DatensatzAlle):	*true*, if all (including non updated) records are to be reported, otherwise *false*

The following example illustrates the retrieval of updated data only:

```
<DataSupplyRequest Sender="AVMS B" TimeStamp="2001-08-08T08:01:00">
      <AllData>false</AllData>
</DataSupplyRequest>
```

If *AllData* is set to *true*, then the system transmits not only the data that has been updated since the last request but all records of all active subscriptions.

Implementation hint:

If a DataSupplyRequest occurs without at least one subscription established before, the server has to send an error message. An empty *DataSupplyAnswer* is not allowed.

5.1.4.2 DataSupplyAnswer

The server responds with the updated records by means of a *DataSupplyAnswer* message. The content is service-specific.

The "PendingData" element indicates whether the content of *DataSupplyAnswer* contains all updated data or whether for technical reasons the transmission has been split into several packets. The data can be retrieved by the data consumers with several *DataSupplyRequests*. In the last data packet, the *PendingData* element is set to "*false*". Contrary to the standard behaviour of optional fields, *PendingData* has default value "*false*". A missing *PendingData* element indicates that the data transmission occurred completely within the packet.

The data of a subscription must be completely included within one data packet. The splitting of subscription data is not permitted.

Definition of *DataSupplyAnswer (DatenAbrufenAntwort)*:

Acknowledge (Bestaetigung):	(sub-element) Contains information for error handling
PendingData (WeitereDaten):	(optional, default) "true" if other data can be polled, otherwise "false" (default).
FeederMessage (Zubringernachricht):	(sub-element, multiple) Contains feeder messages in the connection protection service
FetcherMessage (Abbringernachricht):	(sub-element, multiple) Contains fetcher messages in the connection protection service.
DISMessage (AZBNachricht):	(sub-element, multiple) Contains messages about foreign vehicles approaching a display area
VISMessage (VISNachricht):	(sub-element, multiple) Contains information about trips that are to be visualised in a foreign control centre
GMSMessage (ANDNachricht):	(sub-element, multiple) Contains information on current operational events that are to be made known to the dispatchers in a foreign control centre

With *DataSupplyAnswer* the server receives authorization to reset the update flags of the subscription for which data has been transmitted. A renewed *DataSupplyRequest* would then not be answered with the data of the recently polled subscription.

The following example shows (independent of service) a possible server reply to the above request:

```
<DataSupplyAnswer>
      <Acknowledge
            TimeStamp="2001-08-08T08:01:10"
            Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <ElementXYZ TimeStamp="2001-08-08T08:00:00">
            <SubElement1> ... </SubElement1>
            <SubElement2> ... </SubElement2>
            <SubElement3> ... </SubElement3>
            </ElementXYZ>
      <ElementXYZ TimeStamp="2001-08-08T07:59:00">
            <SubElement1> ... </SubElement1>
            <SubElement2> ... </SubElement2>
            <SubElement3> ... </SubElement3>
            </ElementXYZ>
</DataSupplyAnswer>
```

5.1.5 Deleting Data Subscriptions (*DeleteSubscription/DeleteSubscriptionsAll*)

Subscriptions are automatically deleted by the server after expiration of their validity. If a subscription is to be deleted by a client before expiration of validity, a new *SubscriptionRequest* must be triggered, containing sub-elements of type *DeleteSubscription* (5.1.2), each of which deletes a subscription.

The following example deletes subscriptions “12” and “5”:

```
<SubscriptionRequest Sender="AVMS B" TimeStamp="2001-08-08T08:10:00">
      <DeleteSubscription>12</DeleteSubscription>
      <DeleteSubscription>5</DeleteSubscription>
</SubscriptionRequest>
```

To delete all subscriptions of a service, replace the *DeleteSubscription* sub-elements with one *DeleteSubscriptionsAll* element with the value *true*:

```
<SubscriptionRequest Sender="AVMS B" TimeStamp="2001-08-08T08:10:00">
      <DeleteSubscriptionsAll>true</DeleteSubscriptionsAll>
</SubscriptionRequest>
```

5.1.6 Reset after Interruption

A break in the data connection can be determined by timeout messages of the HTTP protocol. The currently recognised error statuses are as follows:

Lost message	**by**	**Status / action**
SubscriptionRequest	Server	Reply missing; client must send request again
SubscriptionReply	Client	Subscription is set up, client overwrites with renewed request
DataReadyRequest	Client	As the reply is missing, the server sends again
DataReadyAnswer	Server	Renewed transmission of the request, until reply is received from the client
DataSupplyRequest	Server	Client receives no reply. It must assume the reply has been lost (worst case) and request all data again (*AllData*).
DataSupplyAnswer	Client	Data lost, renewed polling not possible because the server has reset the update flag of the subscription. Client must

Lost message	by	Status / action
		initiate an *AllData* request.
DeleteSubscription	Server	Client retransmits message until it receives a reply or there is an error message concerning an unknown *SubscriptionID*.
StatusRequest	Both	Reply missing, sender retransmits or assumes that the connection has been down a long time. Service is no longer available.
StatusReply	Both	See *StatusRequest*.

Table 2: Error statuses and actions in case of connection break

5.1.7 Reset after Crash

If the client loses its subscription data, for example after a crash, the subscriptions must be recreated. The first step is to delete all subscriptions on the server (*DeleteSubscriptionsAll*) and then to create them again.

If the server loses its subscription data, it is not immediately obvious to the client. *DataReadyRequests* are missing, but the client is unable to distinguish this from normal operation and with that is unable to detect a server crash. To be able to recognise this situation, additional cyclic *StatusRequests* (5.1.8) need to be sent to the server. Within the *StatusReply* (5.1.8) the server specifies the time stamp of the service start. If the start of the service is after the time at which the subscription was set up, loss of the subscription must be assumed. The procedure is now as for client data loss – delete and recreate all subscriptions.

5.1.8 Alive Handling/

Status polling is used to establish the availability of the services. A separate information channel is used (target URL status.xml), which must be made available by each service.

5.1.8.1 StatusRequest

If the client wishes to establish whether the service is still “alive”, it sends a *StatusRequest* to the server and waits for the reply (*StatusReply*).

Definition of *StatusRequest (StatusAnfrage)*:	
Sender (Sender):	(attribute) Control centre code of the requesting system
TimeStamp (Zst):	(attribute) Time stamp of generation of request

5.1.8.2 StatusReply

If a positive response is received, the service is available. The *StatusReply* describes the total availability of all information channels of a service. If a single channel is faulty, the entire service is no longer available.

Definition of *StatusReply (StatusAntwort)*:

Status (Status):	(sub-element) Indicates whether the service is available
DataAvailable (DatenBereit):	If *true*, the data is ready to be retrieved
StartServiceTimeStamp: *(StartDienstZst)*	Specifies the time of the start of the service. If the service is not available, any value may be given here

In addition to the time stamp, the *Status* element only contains a *Result* element, which states *true* if the service is available.

Definition of *Status (Status)*:

TimeStamp (Zst):	(attribute) Time stamp of creation of status information
Result (Ergebnis):	(attribute) “ok”, if the service is available, otherwise “notok”.

The *StatusRequest* also enables the client to detect whether a service has been started again and that the subscriptions have been lost. Within *StatusRequest* the server specifies the last start time of the service. A start time after the set-up of a subscription indicates that it has been restarted at some point in between (5.1.7).

Implementation hint: In some cases it might be wise to establish the above described alive handling process also from the perspective of the server. This will not change the described process.

The following example shows a *StatusRequest* with a corresponding (successful) *StatusReply*:

Client request (AVMS A):

```
<StatusRequest Sender="AVMS A" TimeStamp="2002-02-14T14:03:49"/>
```

Server response: Service available, data waiting to be retrieved:

```
<StatusReply>
      <Status TimeStamp="2002-04-02T14:00:00" Result="ok"/>
      <DataAvailable>true</DataAvailable>
      <StartServiceTimeStamp>2002-04-02T06:00:00</StartServiceTimeStamp>
</StatusReply>
```

5.2 Http Link

5.2.1 Procedure

Message exchange via HTTP is achieved using the POST method. This allows data to be transmitted from a client to a server in the form of requests. The server transmits its data in the reply to the request.

Example of HTTP-POST:

```
POST /controlsystem1/ans/status.xml HTTP/1.1
Host: test1:1111
Content-Type: text/xml
Charset="iso-8859-1"
Content-Length: 64

<StatusRequest Sender="AVMS A" TimeStamp="2002-02-14T14:03:49"/>
```

Example of an HTTP response:

```
HTTP/1.1 200 OK
Content-Type: text/xml Charset="iso-8859-1"
Content-Length: 87

<StatusReply>
      <Status TimeStamp="2002-04-02T14:00:00" Result="ok"/>
      <DataAvailable>true</DataAvailable>
      <StartServiceTimeStamp>2002-04-02T06:00:00</StartServiceTimeStamp>
</StatusReply>
```

5.2.2 Character Set

The ISO-8859-1 character set is used exclusively.

5.2.3 Service Codes

The so-called “service” represents the provision of specific data and its processing involving multiple control systems.

The online interface currently supports the following services:

Service	Code	Description
Connection protection reference data service	cpref	On the server side, makes the feeder planning data available. On the client side, this information is processed within the connection protection
Connection protection process data service	cp	On the server side, makes the current real-time feeder data available. On the client side, this information is processed within the connection protection
Passenger information reference data service	dpiref	On the server side, provides departure tables for DPI displays supplied with reference data.
Passenger information process data service	dpi	On the server side, provides the data for passenger information. This data is then shown on the corresponding displays on the client side.

Visualisation of trips	vis	On the server side, provides the trip data, which is then visualised in the control computer on the client side.
Message service	gms	On the server side provides text messages

Table 3: HTTP service codes

5.2.4 Request URL

All requests must be directed at specific target URLs. The request URL depends on the service as well as the type of request.

Name	Request code	Description
Poll status	status.xml	This request tests whether a service on the desired server is responding. The control system code and the service code are transmitted in the reply. This request is also used for cyclic connection monitoring.
Manage data subscription	subscription.xml	With this request it is possible to retrieve online data from the desired control system or to delete existing subscriptions. The reply is either confirmation of acceptance of the request or in case of error, a corresponding error message.
Report data ready	dataready.xml	With this request the readiness of data is reported to a partner system. The partner system then initiates transmission of the data with a "transmit data" request. The reply is either confirmation of acceptance of the request or an error message.
Poll data	polldata.xml	With this request it is possible to poll online data. As a reply, the ready data is transmitted or an error message.

Table 4: HTTP request URL

The URL, to which the above requests are transmitted, is defined as follows:

```
HTTP_URL = "http:" "//"  host [ ":" port ] abs_path
```

`"http:" "//"`	Name of the protocol
`host`	Denotes the HTTP server at which the request is directed.
`":"port`	Specifies the port via which the TCP/IP connection is to be made. Not required if the default port (80) is to be used
`abs_path`	Specifies the request path.

Definition of abs_path for the online interface:

```
abs_path = "/" control centre code "/" service code "/" request code
```

The control centre code must be defined by the interface partners. The specification of the foreign control centre code within the path facilitates specific implementations of the interface, depending on the partner. The service code is established during implementation of a service and the request code can be taken from the above table.

The requests for the various services therefore differ in terms of URL path. The requests are always interpreted by the service to which they have been directed. Cross-service requests are not possible.

Example of a status request (port 8080) to the connection protection service:

`http://serverhost:8080/foreigncontrolcentrecode/ans/status.xml`

5.2.5 Error Handling

The protocol-specific status codes are supported at the level of the HTTP protocol. The most important codes are listed in the following table:

Code	Short name	Description
200	OK	Request successful
400	Bad Request	The server does not "understand" the request. The client should not repeat the request.
401	Unauthorized	User name and password are required for the request.
403	Forbidden	The server "understands" the request, but cannot carry it out.
404	Not Found	The requested URL was not found
408	Request Timeout	Server not responding

Table 5: HTTP error messages

For the online interface, it is sufficient for status code 200 (OK) to be distinguished from the rest on the client side. This indicates successful reception of the request. All other codes supplied by the HTTP server must be interpreted as a rejection of the request. The supplied status code supplies more information on the reason for the rejection.

If the server does not respond within the given timeout, this must also be interpreted as a rejection of the request.

Errors at the technical level are registered via defined fields in the context of message confirmation (*ErrorNumber*, *Errortext*).

5.3 Security

AVMS systems are usually incorporated into a company network. As the connection of internal systems via the internet is considered a security risk, a VPN (virtual private network) is recommended for handling the data traffic. This connects computers or sub-networks of different company networks, resulting in a virtual secure network, to which only the connected computers have access. Within this virtual network, the data is transmitted in encrypted format and can therefore be passed on via unsecured connections (the internet).

6 Interface Description for “Technical Services”

The technical services are based on the generic subscription concept. They complement the requests and replies with the technical data that is required to carry out the services. With regard to the subscriptions, the processing of messages is standardised.

The following sections describe the technical services of connection protection (CP), dynamic passenger information (DPI), visualisation of foreign vehicles (VIS) and the general message service (GMS).

6.1 General Terms

There are certain fundamental problems when exchanging data between different operations: How is it possible to achieve common understanding of the objects (e.g. stops)? How can they be represented in accordance with a common reference? How long do these objects “live”, i.e. what are the time considerations for data management? This chapter describes the fundamental definitions, used as a basis for the subsequent modelling. The results are reflected in the data definitions in the AVMS data storage systems. This data is hereafter referred to as meta data.

6.1.1 Operating Days

Data exchange is carried out on the basis of the operating data provided in the AVMS. As the storage times for planning data can differ from operation to operation, this specification is restricted to an exchange of data relating to the current day only.

The problem here is that the operating times of the AVMS systems of different operators are usually different. Some systems have breaks, in which the systems are reinitialised and data is transferred from the schedule planning. Other systems work on a 24-hour basis, without a break. It should be noted however, that it is not absolutely necessary to a have a clear cut changeover point for the data supply. Several schedule versions can co-exist within a defined transition time. From this, we can deduce that there is no clearly defined time frame in which it is possible to guarantee that both participant systems can have full data availability. The rule represents partial availability, in which a system cannot always provide all requested data. This is particularly relevant to the exchange of reference data, but it is also impossible to guarantee availability of spontaneously requested process data.

For this reason, the specification outlines a request time period, which defines a time frame for requests and a feedback message that enables the end of the data horizon to be reported. This means the consumer knows that the request cannot be answered (completely), or that it must be repeated at a later time.

6.1.2 Date and Time Format

Every piece of time information is related to UTC (Coordinated Universal Time). Deviations from this time zone are coded in accordance with ISO 8601 (e.g.: 2000-04-07T18:39:00+01:00).

Without a specification of time difference, the time is already in UTC. In this situation there may also be a subsequent Z (2002-04-30T12:00:00 corresponds to 2002-04-30T12:00:00Z). In other words, the first 19 characters are obligatory and correspond to local time or UTC.

Time units less than one second are ignored.

This method of time representation avoids any problems with summer/winter changeover.

6.1.3 Control Centre Code

In order to be able to differentiate between the messages from different communication partners within a service, each message contains a unique control centre code (attribute *Sender*) relating to the requesting system. All further bilaterally agreed meta data refer to this control centre code.

6.1.4 Location References

In addition to the current planning data, two systems offering inter-operational connection protection need additional planning data that facilitates common understanding of the respective location references. These respective internal location codes (stop reference, display reference, etc.) could be managed on a mutual basis.

There is however a fundamental disadvantage with this kind of definition. The direct linking of one systems location codes with those of another system means that the data supplies are also coupled. Changes to the master data can very quickly lead to inconsistencies.

To avoid this, an independent location code is introduced, which effectively decouples the data storage systems. The basic idea is that with the agreement of “common” names / codes, direct coupling becomes obsolete. There is only partial linking of the data with the “meta” data. The modelling can be based on location points (e.g. stops, masts) or location areas (stop areas). The most suitable model depends on the AVMS data model as well as the local situations (changeover times).

There are no location codes associated with the message service but here too data is requested on the basis of bilaterally agreed IDs that decouple the systems. These designators are called *channels* and split the volume of possible messages into classes. With that, the channel represents the equivalent of the location reference.

The following definitions must be specified in the respective services:

Service	**Location name**	**Code name**
Connection protection	Connection area	CPIID
Dynamic passenger information	Display area	DISID
Visualisation	Visualisation area	VISID
Message service	Channel	ChannelID

Table 6: Location codes in the technical services

The common names can be agreed within the scope of a scheduling conference or other similar committee. The agreement should be valid in the mid to long term, to minimise the necessity of data changes.

The agreements can be exclusively bilateral. Within the data exchange, it is always possible to determine the reference to the communication partner. This means that all names need only be uniquely related to the respective communication partner. A regional agreement of terms between several operators could be useful but is not essential.

All reference data within this specification must be linked with the internal operating data via this concept. There is no need for a common data supply that would exceed this meta data.

Note: In order to be able to manage changes to the data that occur before usage, data storage also demands a means of defining different versions of the meta data, which is coordinated for every communication partner.

6.1.5 Trip References (TripID)

The referencing of foreign trips presents a similar problem to that of location references. The operational terms in use, such as line / block / run do not have common semantics. This means there is also no guarantee of the required uniqueness.

Within an AVMS, there is therefore an internal designator that uniquely references the trip (within the given data horizon). However, because the data horizon of the foreign operator remains unknown, it is not possible to eliminate the possibility of two trips of a foreign operator with the same internal designator within the same operating day (in the foreign AVMS they are different operating days).
To use the AVMS internal trip designator as an inter-operational meta datum therefore, it is necessary to complement it with the operating day code in the interface:

Definition of *TripID (FahrtID)*:

TripName (Fahrtbezeichner):	Unique reference to the trip depending on the operating day.
DayType (Diensttag):	Denotes the operational day on which the trip takes place

It is irrelevant how the operating day is defined internally. What is important is the uniqueness of the combined *TripID* element.

6.1.6 Line and Direction References

In order to keep the data storage independent of line and direction references as well, separate, independent *LineID* and *DirectionID* elements are also used in the interface. These are agreed bilaterally.

Within the data management, it is necessary to create a reference between the *LineID* or *DirectionID* and the operators own line or direction. Identifiers of foreign operators are included in the connection definitions (6.2.2).

6.1.7 Product Types

Within the interface, product types serve to classify the trips. Conceivable products include “Intercity train”, “express bus”, etc. As yet, there is no common standard for product classification.

In order to be able to present common product names (texts, icons) to the passenger in the case of inter-operational connection protection and dynamic passenger information, meta data management is again required here. It is necessary to agree *ProductID* and the associated quality classes. Similar to all other meta data, the agreement is again bilateral.

As the use of ProductID is optional, it is also possible to avoid the agreement of inter-operational product types.

6.1.8 Circular Trips

The announcement of planning and process data requires unique referencing of the arrival of a feeder at a stop. However, the combination of *TripID* and *CPIID* is not always unique. If a trip serves a particular stop more than once, it is not possible to make the differentiation between the first and second arrival. For this, we use the so-called stop sequence counter (*StopSeqCount*) as the definitive criterion.

The stop sequence counter is a positive integer, which orders the multiple arrivals of a trip at a CPI into an ascending sequence. The StopSeqCount need not increase sequentially, but must increase in a strictly monotonous way.

Data producing systems, which use other methods for differentiating circular trips (planned arrival time, route point counter), can use this data directly (coded as an integer) as long as the monotony is observed.

Circular trips for the fetchers are not considered.

6.1.9 Service Types

In order to be able to represent operational services such as school bus, bus with disabled facilities, etc. both systems must include references to common inter-operational meta data. This is achieved within the scope of the operational data management:

Own service feature	Inter-operational service feature	Foreign operator code
School trip	SCH	AVMS B
Disabled access	DBA	AVMS B
Pull-in journey	PUL	AVMS B
Museum trip	MUS	AVMS B

Table 7: Definition of inter-operational service features

6.1.10 Errors at the Technical Service Level

In addition to the general errors, which occur as a result of the unavailability of client/server/network and with that at the HTTP level, there are also errors within the technical message communication. If, for example, a reference to a meta datum (e.g. *CPIID*) is unavailable, the requested service cannot be carried out and an error message must be returned to the requesting system. This is achieved within the *Acknowledge* element, which is a sub-element of every reply message. It also includes an optional *ErrorNumber* element, which specifies an error category:

Error number	Meaning/cause
0	OK: (no error)
100-199	XML error Examples: XML document not formed correctly, error checking against schema
200-299	Reference data violation Example: Invalid "Sender" attribute in a request
300-399	Other error All other errors based on faulty requests are registered in this category. The request therefor should not be repeated.
400-499	Other answers than faulty requests. Example: The requested data are in a process of editing and therfor not available. A later request may lead to a positive result.

Table 8: Technical error types

The *Errortext* element is also provided for a more accurate, textual description of the error. This description must contain at least the faulty element along with its value. In general this message should be as detailed as possible as it represents the only possibility for trouble shooting.

If an error occurs within a *SubscriptionRequest*, the entire request is invalid and must be rejected. It is neither possible to set up or delete subscriptions if the syntactic or semantic tests fail. The meta data should therefore be checked in advance or a transactional system implemented.

6.1.11 Optional Fields

Within the communication, a lot of information is optional. This means that the receiver cannot count on the availability of this data if the data producing system cannot make it available on the basis of technical or other operational reasons. For this reason, it is not possible to use default settings for optional fields.

Exceptions to this are explained separately (see *Trainset, PendingData)*.

6.2 Connection Protection (REF-CP, CP)

6.2.1 Introduction

Inter-operational connection protection demands that both operators have AVMS systems that provide internal connection functionalities. The data exchange described in this chapter ensures the systems are in a position to receive all the necessary data concerning the feeder vehicles to allow connection monitoring and dispatch to be carried out. The operational methods of dispatch remain unaffected.

The interface provides two services. The reference data connection protection service (REF-CP) provides functions for transferring scheduled arrival times in the connection areas.

The process data connection protection service (CP) offers functions for exchanging real-time data (schedule deviations, effects of dispatch actions). The process data service can be used in two different formats:

1. Trip-based exchange of process data

   Corresponds to the method in VDV 453, version 1.0. This involves an exchange of the process data of previously announced trips. The registration of the trips is achieved via common scheduling or by the use of the reference data connection protection service (REF-CP). This means that the process of agreeing the connections can be started directly after importing the current day's schedule into the AVMS system.The number of connections is therefore known in advance.

2. Time-based connection protection

   This method is new to version 2.0 of VDV 453. It permits connection protection without the previous exchange of reference data. Connection agreements are short-term.

With the transfer protection service, message transmission is defined from the feeder to the receiver (feeder message) as well as – for the process data service – from the receiver to the feeder (receiver message).

1. Feeder messages

All messages that are transmitted from the feeder to the receiver are encapsulated in a feeder message element (6.2.4.3). The feeder message is used in the reference data service and the process data service.

Within the context of the transfer protection reference data service, the feeder message consists exclusively of schedule information (6.2.3.3 Transferring Area Schedules (CPISchedule)~~Transferring Area Schedules (CPISchedule)~~.

In the process data service the feeder message may consist of any combination of schedule status information (6.2.4.3.1 Transferring Connection Data (CPIDeviation)) and trip cancellation messages (6.2.4.3.2 Feeder Loss (CPITripDelete).

2. Receiver messages

In addition to the messages that are communicated from the feeder to the receiver, there are additional messages in the process data service, which travels in the opposite direction from the receiver to the feeder. It is used for passenger information in the feeder trips. All reverse messages are optional. Whether or not a system sends these messages is subject to interoperational agreement. A differentiation on the basis of individual partners is regulated by the management of meta data.

6.2.2 Operational Data Supply and Management

The simplest case of exchanging reference connection protection data only requires the specified supply with common connection area codes (*CPIID*). An allocation table is created, assigning the abstract location codes (connection areas) to the internal location codes (stops or areas) (6.1.3). On the side of the fetcher system it is also necessary to provide the changeover time.

This type of definition could look as follows in both systems:

CPIID	Foreign operator code	Internal stop code
12345	AVMS B	3642
12346	AVMS B	4564
35678	AVMS B	7765

Table 9: Assigning connection location codes for the feeder (AVMS A)

CPIID	Foreign operator code	Internal stop code	Changeover time
12345	AVMS A	2345	0
12346	AVMS A	3687	0
35678	AVMS A	7566	1

Table 10: Assigning connection location codes for the fetcher (AVMS B)

With that, it is already possible to make requests for feeder data. However, it is not at this stage possible to make selections based on specific connection relationships of lines, directions or time of day. This requires extended data management, which defines additional connection relationships. Data management is carried out from the point of view of the fetcher.

Foreign operator code	CPIID	LineID (Fdr)	DirectionID (Fdr)	LineID (Ftr)	DirectionID (Ftr)	Time
AVMS A	12345	10	Zoo	12	CSt.	10:00-12:00
AVMS A	12345	10	Station	12	CSt.	10:00-14:00

Table 11: Definition of connection relationships for the fetcher

This defines that trips in two directions (Zoo, Station) of one line (10) of foreign operator AVMS A are potential feeders, but only between 10:00 a.m. and 12:00 p.m. or 10:00 a.m. and 2:00 p.m. respectively and only if the fetcher vehicle of the actual operator is travelling on line 12 in direction CSt.

From the above table it can be seen that in addition to the *CPIID*, the foreign line or direction must also be managed as meta data. Even the operators own line and direction information needs to be managed as meta data, as for return channel messages or subscriptions, it is necessary to provide fetcher information.

On the feeder side, it is not necessary to manage the fetcher line or direction meta data.

If other data is to be incorporated into the process of finding connections (service characteristics, product type), these criteria must be defined when defining the connection relationships. It should also be noted that this information is optional.

Product types (*ProductID*) must also be agreed between the operators.

6.2.3 Reference Data Service (REF-CP)

The reference data service is used for the exchange of planning data for potential feeder trips. The service is specific to a location, i.e. all requests and replies refer to concrete transfer protection areas.

The exchange of reference data is only meaningful in the context of trip-specific transfer protection. Trip-specific transfer protection demands knowledge of the feeder trips. This can be achieved via a common schedule supply or via exchange using the reference data service.

The reference data service is initiated by the receiver. It requests feeder trips for a defined transfer protection area. Assuming the existence of operational meta data management, the request can be restricted to foreign routes and directions.

The feeder responds to the request by supplying departure tables for the requested transfer protection areas. The TripID supplied for every trip is later used for referencing in the subscriptions of the transfer protection process data service.

With that the reference data service satisfies two purposes:

- Preparation for the subscription of process data (trip specific)
- Finding transfers in the run up

Any changes to the schedule caused by dispatch actions are also communicated in the context of the subscription process. This function is optional on the server side. Clients however must always be able to formally accept and confirm updates.

6.2.3.1 Data Exchange

The fetcher initiates the reference data service. For every defined connection relationship (Table 11) the fetcher system generates a *SubscriptionRequest* (and with that a *SubscriptionID* unique within the service). The request consists of the *CPIID* of the connection location and the additional (optional) filters of *line* and *direction*. It is possible to define several requests within one *SubscriptionRequest*.

The technical data of the request is encapsulated in a so-called *CPIRefSubscription* element, which represents the service-specific frame. The request time frame usually covers the entire availability horizon of the proprietary system (to the end of the operating day, or the end of the operating day on the fetcher line).

After receiving the message and checking the validity of the meta data, the feeder system returns an *Acknowledge* as confirmation.

The feeder system then searches its own planning data (including current dispatch actions) and compiles the departure tables. The readiness of the data is signalled to the fetcher with a corresponding *DataAvailable* message.

The fetcher can now request all modified data (*DataSupplyRequest* message). The feeder system replies with a *DataSupplyAnswer* message, which contains the technical data (departure tables).

6.2.3.1.1 Availability Horizon

As the end of the operating day of one AVMS system does not usually coincide with the end of the operating day of another AVMS system, it must be assumed that it is impossible to make data available for the entire request time frame. To ensure the data consumer (fetcher) is aware of this situation, there is an optional *DataValidUntil* element defined in the *Acknowledge* message of the *CPIRefSubscription* request. If the feeder receives a request, which goes beyond its data availability horizon, it can enter the end of the horizon here. If the *DataValidUntil* element is missing, the fetcher can assume that its request can be fully satisfied.

If the request time frame lies completely outside the data horizon of the producer, this situation is signalled by means of a value in *DataValidUntil,* which lies before the time of the request.

6.2.3.1.2 Updating

After submitting a request, the fetcher receives the relevant response in the form of departure tables. If these tables change, new *DataReadyRequest* messages are sent to the fetchers to indicate that the data has changed.

Communicated changes involve additional trips only. The reason is unimportant (reinforcement trip, special purpose vehicle, diversions, etc.)

Changes to the trip itself (start stop, departure time, mode of transport, product) remain hidden, or are communicated within the context of process data exchange. Trips that no longer reach the connection area as a result of a dispatch action are removed from the connection protection via an explicit message (*CPITripDelete*).

Replacement trips for any that have failed should be transparent (recommendation). It is simply the same trip with a different vehicle.

On the server side updates are optional. Clients on the other hand must always be able to receive updates.

6.2.3.2 Requesting Area Schedules (CPIRefSubscription)

The subscription of schedule data is achieved via a *SubscriptionRequest* with one or more embedded elements of type *CPIRefSubscription*.

The *CPIRefSubscription* elements specify the connection area, optional filter criteria as well as the request time frame for which data is to be supplied.

Definition of *CPIRefSubscription (AboASBRef)*:

SubscriptionID(AboID) (attribute):	The SubscriptionID references the subscription of feeder data created by the request. The SubscriptionID is specified by the fetcher system.
ValidUntilTimeStamp(*VerfallZst*) (attribute):	Specifies the time to which the subscription is valid
*CPIID (*ASBID):	References the connection area
LineID (LinienID):	(optional) Filter for the feeder line, which is to supply data
DirectionID(RichtungsID):	(optional) Filter for the feeder direction, for which data is to be supplied.
EarliestArrivalTime(FruehesteAnkunftsszeit):	Defines the start of the time frame for which data is to be supplied. The reference is the arrival time of the feeder at the connection area.
LatestArrivalTime(SpaetesteAnkunftsszeit):	Defines the end of the time frame, for which data is to be supplied. The reference is the arrival time of the feeder at the connection area.

LineID and *DirectionID* are independently optional. It is therefore possible to specify a request that only includes a direction filter without a line filter.

The *EarliestArrivalTime* element should not be set to a value before the start of the actual data horizon, and *LatestArrivalTime* should not be after the end of the horizon. The *ValidUntilTimeStamp* attribute should be equal to or later than *LatestArrivalTime.*

The following example describes a reference data request (“AVMS A” is the fetcher) for connection area “12345”. The requesting system only requires feeder data from line 10 in direction “Zoo”. The fetcher AVMS, which makes the request, has a data horizon of 5:00 a.m. to 11:00 p.m., 8.8.2001. The feeder AVMS only has a data horizon up to 10:00 p.m.

```
<SubscriptionRequest Sender="AVMS A" TimeStamp="2001-08-08T05:00:00">
      <CPIRefSubscription SubscriptionID="25" ValidUntilTimeStamp="2001-08-
09T00:00:00">
      <CPIID>12345</CPIID>
      <LineID>10</LineID>
      <DirectionID>Zoo</DirectionID>
      <EarliestArrivalTime>
            2001-08-08T05:00:00
      </EarliestArrivalTime>
      <LatestArrivalTime>
            2001-08-08T23:00:00
      </LatestArrivalTime>
      </CPIRefSubscription>
</SubscriptionRequest>
```

An *Acknowledge* reply is received from the feeder within the *SubscriptionReply* message. The restricted data horizon is signalled in *DataValidUntil*:

```
<SubscriptionReply>
      <Acknowledge TimeStamp="2001-08-08T05:00:05" Result="ok"
            ErrorNumber="0">
            <DataValidUntil>2001-08-08T22:00:00</DataValidUntil>
      </Acknowledge>
</SubscriptionReply>
```

6.2.3.3 Transferring Area Schedules (CPISchedule)

Once the reference data subscriptions are set up (6.2.3.2), the feeder system transmits the departure tables and signals their readiness with the initial *DataReadyRequest* message:

```
<DataReadyRequest Sender="AVMS B" TimeStamp="2001-08-08T05:01:00">
</DataReadyRequest>
```

Reception is acknowledged by the fetcher with an *Acknowledge* within the *DataReadyAnswer* message:

```
<DataReadyAnswer>
            <Acknowledge TimeStamp="2001-08-08T05:01:01" Result="ok" Error-
                  Number="0"></Acknowledge>
</DataReadyAnswer>
```

The fetcher then requests the data with a *DataSupplyRequest*:

```
<DataSupplyRequest Sender="AVMS A" TimeStamp="2001-08-08T05:01:05">
</DataSupplyRequest>
```

The feeder responds to this message with the requested data within elements of type *CPISchedule*. The *CPISchedule* is in turn a sub-element of the so-called *FeederMessage (0)* and corresponds to a concrete arrival at a connection area. A *FeederMessage (6.2.4.3)* is directly assigned to a subscription.

The arrival table therefore consists of a list of several *CPISchedule* elements.

Definition of *CPISchedule (ASBFahrplan)*:	
TimeStamp(Zst):	(attribute) Time the schedule data changed
CPIID(ASBID):	References the connection area
StopSeqCoun(HstSeqZaehler)t:	Passage counter for circular trip detection. Value increases with the number of passages.
TripID (FahrtID):	(sub-element) References a feeder trip
LineID(LinienID):	Line code of the feeder
LineText (LinienText):	Line description (for passenger)
DirectionID(RichtungsID):	Indicates direction of feeder
DirectionText(RichtungsText):	Description of direction (for passenger)
ScheduledCPIArrivalTime(AnkunftszeitASBPlan):	Scheduled arrival time of the feeder at the connection area
TripInfo(FahrtInfo):	(sub-element, optional) Additional information on the feeder trip

6.2.3.3.1 Additional information to the trip (*TripInfo*)

Information in the *TripInfo* element has no functionality within the context of connection protection. It is therefore all optional with the main purpose of providing extra information to the dispatcher, which may be useful in the course of telephone conversations with other control centres or for logging purposes.

Definition of *TripInfo (FahrtInfo)*:	
VehicleID (FahrzeugID):	(optional) Internal AVMS name for the vehicle
LineNumber (LinienNr):	(optional) Internal AVMS line number of the trip. May differ from the LineID (meta data)
BlockNumber (UmlaufNr):	(optional) Internal AVMS block number of the trip
RunNumber (KursNr):	(optional) Internal AVMS run number of the trip
DepartureStopLong (StartHstLang):	(optional) Full name of the start stop
DestinationStopLong (ZielHstLang):	(optional) Full name of the end stop
DepartureStop (StartHst):	(optional) Short name of the start stop
DestinationStop (ZielHst):	(optional) Short name of the end stop
DepartureTimeStartStop (AbfahrtszeitStartHst):	(optional) Planned departure time of the trip at the start stop
ArrivalTimeDestinationStop (AnkunftszeitZielHst):	(optional) Planned arrival time of the trip at the destination stop
Operator (Betreiber):	(optional) Name of the (sub) contractor operating the trip
ProductID (ProduktID):	(optional) Unique product reference
ServiceAttribute (ServiceMerkmal):	(optional, multiple) Specifies that the trip has the agreed service characteristic (ex. Vehicle with low floor for disabled persons)
Direct Call (Direktruf)	(sub-element, optional) Information about direct communication with the vehicle

A typical response to the request in 6.2.3.2 could read:

```
<DataSupplyAnswer>
      <Acknowledge
            TimeStamp="2001-08-08T05:01:07"
            Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <FeederMessage SubscriptionID="25">
            <CPISchedule TimeStamp="2001-08-08T05:01:05">
                  <CPIID>12345</CPIID>
                  <TripID>
                        <TripName>64356</TripName>
                        <DayType>2001-08-08</DayType>
                  </TripID>
                  <LineID>10</LineID>
                  <LineText>X10</LineID>
                  <DirectionID>Zoo</DirectionID>
                  <DirectionText>Zoological Garden</DirectionID>
                  <ScheduledCPIArrivalTime>
                        2001-08-08T06:10:00
                  </ScheduledCPIArrivalTime>
                  <StopSeqCount>1</StopSeqCount>
                  <TripInfo>
                        <VehicleID>34567</VehicleID>
                        <LineNumber>10</LineNumber>
                        <ServiceAttribute>School bus</ServiceAttribute>
                        <ServiceAttribute>
                              Disabled access
                        </ServiceAttribute>
                        <ServiceAttribute>Air con</ServiceAttribute>
                        <Operator>BVG</Operator>
                        <ProductID>Express bus</ProductID>
                        <DirectCall>
                              <Telephonnumber>
                                    +4917633445566
                              </Telephonnumber>
                        </DirectCall>
                  </TripInfo>
            </CPISchedule>
            <CPISchedule TimeStamp="2001-08-08T05:01:06">
                  <CPIID>12345</CPIID>
                  <TripID>
                        <TripName>44347</TripName>
                        <DayType>2001-08-08</DayType>
                  </TripID>
                  <StopSeqCount>1</StopSeqCount>
                  <LineID>10</LineID>
                  <LineText>X10</LineID>
                  <DirectionID>Zoo</DirectionID>
                  <DirectionText>Zoological Garden</DirectionID>
                  <CPIArrivalTime>2001-08-08T06:20:00</CPIArrivalTime>
            </CPISchedule>
      </FeederMessage>
</DataSupplyAnswer>
```

In this situation exactly two trips are supplied, line X10 direction Zoological Garden with arrival times of 6:10 a.m. and 6:20 a.m.

6.2.3.3.2 Information about direct communication to the vehicle (DirectCall)

The element DirectCall provides information about how to communicate with the conductor of the vehicle from the view of the control center.

Telephonenumber (Telefonnummer):	(optional) Telephonenumber (international Format)
IP-Adress (IP-Adresse)	(optional) IP-Adress or IP-Number

6.2.4 Process Data Service

The connection protection process data service is used to exchange current schedule deviations and events that are relevant to the passenger in connection with dispatch actions relating to connections. Within the process data service, it is necessary to make a distinction between the following procedures:

→ Trip-related connection protection

→ Time-related connection protection

The trip related connection protection facilitates the request and exchange of data for exactly one, previously known feeder.

Time related connection protection is used to request and exchange data from one or more vehicles, which represent potential feeders within a connection area. The restriction of the data is achieved by specifying line, direction and a time window, in which the arrivals of the approaching trips must lie.

Both procedures are implemented within this service using different filters. The exchange of data is achieved with identical messages and data structures. Differences in the behaviour and results arise from the different requests of the two methods.

6.2.4.1 Data Exchange

Data exchange in the process data service is initiated by the fetcher system. It creates a subscription, which specifies the trip data that it requires. This can either be an individual trip (trip-based connection protection) or a batch of trips which are predicted to reach the connection area within the given time window.

If the subscription is set up on the feeder side, it responds immediately with the current data for the subscription (*CPIDeviation*). This data represents predictions, or in the absence of predictions, planning data.

If the process data of the trips assigned to the subscription changes (schedule deviation of the feeders, dispatch actions), the feeder registers the existence of new data (*DataReadyRequest*).

The fetcher can now explicitly request the updated data only (*DataSupplyRequest*). After a loss of data, there is also the possibility of requesting the entire data set again (*AllData*).

There is a further possibility of reversing the flow of information and within the scope of a subscription communicating messages from the fetcher to the feeder (6.2.4.4).

6.2.4.1.1 Updating / Hysteresis

In the context of connection protection, a process data subscription is viewed as having changed if at least one of the following pieces of information has changed:

→ Schedule deviation

→ Connection cannot take place (feeder does not reach connection area or reaches it too late)

→ Arrival at connection area

A change remains active until the data assigned to the subscription is explicitly retrieved, which resets the status.

In order to avoid constant updates on the basis of minimal changes to the predicted arrival time, a corresponding hysteresis value can be assigned to the feeder within the subscription. This defines the time span, after which the system regards the change 1significant enough to demand communication. With a hysteresis value of 120 seconds, for example, changes in deviations of more than minutes are reported. The hysteresis is based on the last transmitted arrival time. However, the hysteresis value is only a recommendation and can also be ignored by the data producing system.

Updating ends at expiration (expiry timestamp exceeded) or after specific deletion of the subscription.

6.2.4.1.2 Preview Time

In addition to the hysteresis, in the case of Trip-based exchange of process data there is another parameter that restricts the transmission of data in response to a subscription, the preview time. The preview time defines the time window before the actual arrival of the feeder in the connection area within which its schedule deviation information should be transmitted. If the vehicle is not on route at this time, the information transmitted relates to the planning data.

The preview time is optional. If not specified, the time points of the subscription are valid (trip-based), or *EarliestPossibleArrivalTime*.

6.2.4.2 Requesting Connection Data (CPISubscription)

Generally, the trip based retrieval of process data occurs as soon as possible after receiving the schedule data via the reference data service.

Communication is initiated by the fetcher. It creates a *SubscriptionRequest* with an embedded *CPISubscription* element. The *CPISubscription* element in turn may contain the *TripFilter* and *TimeFilter* sub-elements, which defines the trip- and time-related filters. Either one *TimeFilter* or one or more *TripFilter* elements may be defined within *CPISubscription*, but not both. At least one Filter has to be defined.

Definition of *CPISubscription (AboASB)*:

SubscriptionID (*AboID*)	(attribute): The SubscriptionID references the subscription for feeder data created by the request. The SubscriptionID is assigned by the fetcher system.
ValidUntilTimeStamp (*VerfallZs*)	(attribute): Specifies the time to which a subscription is valid
CPIID (ASBID):	References the connection area
TripFilter (Fahrtfilter):	(alternative, multiple) Sub-element, which defines the data request for a feeder trip.
TimeFilter (Zeitfilter):	(alternative) Sub-element, which defines the data request time window.
Hysteresis (Hysterese):	Desired value of change in seconds, after which an update is to be communicated to the fetcher.
FetcherInfo (AbbringerInfo):	(optional) References the fetcher trip.

6.2.4.2.1 Trip Filter

The element *TripFilter* only specifies the feeder as well as the planned arrival time (circular trip problem, 6.1.8). Both at the same time are not permitted. At least one filter must be specified. The schedule statuses of the requested vehicles are sent once the *PreviewTime* has been reached and after every change (*CPIDeviation*).

Definition of *TripFilter (FahrtFilter)*:

TripID (FahrtID):	(sub-element) Unique code for the trip.
StopSeqCount (HstSeqZaehler):	(optional) Passage counter for circular trip detection. Value increases with the number of passages. Not sequential.
ScheduledCPIArrivalTime (AnkunftszeitASBPlan):	Planned arrival time of the trip in the connection area.
PreviewTime (Vorschauzeit):	(optional) Specifies the time period before the planned arrival at the connection area, after which transmission of the schedule deviation data is to start (in minutes).

In the following example, one operator (AVMS A) requests data relating to feeder vehicle “7748” at connection area “12345” for fetcher trip “2208”. The expiry date has been set to the planned departure time of the fetcher (4:00 p.m.) plus the maximum waiting time of 10 minutes. This guarantees a supply of data even in the case of a severe delay. The transmission of the schedule deviation information should be started 30 minutes before the trip arrives in the connection area.

```
<SubscriptionRequest Sender="AVMS A" TimeStamp="2001-08-08T05:10:00">
      <CPISubscription SubscriptionID="25" ValidUntilTimeStamp="2001-08-
08T16:10:00">
            <CPIID>12345</CPIID>
            <TripFilter>
                  <TripID>
                        <TripName>7748</TripName>
                        <DayType>2001-08-08</DayType>
                  </TripID>
                  <StopSeqCount>1</StopSeqCount>
                  <PreviewTime>30</PreviewTime>
            </TripFilter>
            <Hysteresis>120</Hysteresis>
            <FetcherInfo>
                  <TripID>
                        <TripName>2208</TripName>
                        <DayType>2001-08-08</DayType>
                  </TripID>
                  <LineID>3</LineID>
                  <LineText>3</LineText>
                  <DirectionID>CST</DirectionID>
                  <DirectionText>Central Station</DirectionText>
                  < ScheduledCPIDepartureTime>
                        2001-0808T16:00:00
</ ScheduledCPIDepartureTime>
            </FetcherInfo>
      </CPISubscription>
</SubscriptionRequest>
```

The feeder confirms reception of the message with an *Acknowledge* in *SubscriptionReply*.

6.2.4.2.2 Requesting Time-Related Connection Data (*TimeFilter*)

The requesting of time-related connection subscriptions is similar to the process for trip-based subscriptions. The only difference lies in the specification of exactly one *TimeFilter*. This defines the feeder lines/directions, whose trips are to supply schedule deviations and messages. The filter criteria are optional, i.e. it is possible to specify all directions of a line, all lines in one direction, or all trips approaching a connection area. The two times within the filter define the time window within which the trips must reach the connection area in order to be reported. Trips once transmitted, are also subsequently transmitted, even if they no longer fall within the time window. The transmission of the schedule deviation data starts shortly after the subscription has been set up with the feeder.

Definition of *TimeFilter (Zeitfilter)*:

LineID (LinienID):	(optional) Code for the feeder line, which is to supply feeder data
DirectionID (RichtungsID):	(optional) Direction code of the feeder line, for which data is to be supplied.
EarliestArrivalTime (FruehesteAnkunftszeit):	Start of the arrival time window for which feeder data is to be supplied
LatestArrivalTime (SpaetesteAnkunftszeit):	End of the arrival time window for which feeder data is to be supplied

In the following example, feeder data relating to line 2, direction “Station” is to be retrieved for a trip (AVMS A) approaching a connection area. The system is only to provide data for vehicles that are currently predicted to reach the connection area between 3:50 p.m. and 4:10 p.m.

```
<SubscriptionRequest Sender="AVMS A" TimeStamp="2001-08-08T15:45:00">
      <CPISubscription SubscriptionID="25" ValidUntilTimeStamp="2001-08-
08T16:10:00">
            <CPIID>12345</CPIID>
            <TimeFilter>
```

```
                  <LineID>2</LineID>
                  <DirectionID>Station</DirectionID>
                  <EarliestArrivalTime>
                        2001-08-08T15:50:00
                  </EarliestArrivalTime>
                  <LatestArrivalTime>
                        2001-08-08T16:10:00
                  </LatestArrivalTime>
            </TimeFilter>
            <Hysteresis>120</Hysteresis>
      </CPISubscription>
</SubscriptionRequest>
```

Schedule deviation is only to be sent if the change in deviation exceeds 120 seconds.

6.2.4.2.3 Additional Information for the fetcher (*FetcherInfo*)

The optional element *FetcherInfo* is used for the supply of scheduled information for the display in feeder vehicles (fetcher display).

Definition FetcherInfo (*AbbringerInfo)*:

TripID (FahrtID):	(sub-element) Unique code for the trip.
StopSeqCount (HstSeqZaehler):	(optional) Passage counter for circular trip detection. Value increases with the number of passages. Not sequential.
LineID (LinienID):	(optional) Code for the fetcher line, which is to supply feeder data
LineText (LinienText):	Line description (for passenger)
DirectionID(RichtungsID):	(optional) Filter for the fetcher direction, for which data is to be supplied.
DirectionText(RichtungsText):	Description of direction (for passenger)
ScheduledCPIDepartureTime (AbfahrtsszeitASBPlan):	Planned arrival time of the trip in the connection area.
StopID (HaltID)	(optional) references the stop position within the connection area*StopPositionText (HaltepositionsText)*: (optional) Planned Stop position in the connection area.
TripInfo(FahrtInfo):	(sub-element, optional) Additional information on the fetcher trip

6.2.4.3 Feeder Messages (FeederMessage)

All messages that are transferred from the feeder to the receiver are encapsulated within the FeederMessage element. This element does not have any data elements itself, instead it logically groups all message elements that are transmitted from the feeder to the receiver. The following messages can be reported within the feeder message:

In the reference data service:

- Area schedule data (TPASchedule)

In the process data service:

- Current schedule status (TPAScheduleStatus)
- Cancellation of a trip (TPATripDelete)

Definition FeederMessage (*Zubringernachricht*):

SubscriptionID (AboID):	(attribute) Unique reference for the subscription within the service.
CPISchedule (ASBFAhrplan):	(sub-element, optional, multiple) Contains the data on the schedule of the feeder
CPIDeviation (ASBFahrplanlage):	(sub-element, optional, multiple) Contains the data on the actual schedule status of the feeder
FetcherTripDelete(ASBFahrtLoeschen):	(sub-element, optional, multiple) Reports the cancellation of a feeder trip

6.2.4.3.1 Transferring Connection Data (CPIDeviation)

Once the subscription is set up, the feeder initially transmits the data for the desired trip (trip-based subscription), or the desired time window (time-based subscription). It signals the readiness of data for transmission with a *DataReadyRequest*, whose reception is confirmed by the fetcher with an *Acknowledge* in *DataReadyAnswer* (6.2.3.3). With time-based subscriptions, the time of the initial signalling is expected shortly after setting up the subscription. With trip-related subscriptions, the first announcement is expected after the *PreviewTime* has been reached.

The fetcher now requests the data of all subscriptions that has changed since the last *DataReadyRequest*:

```
<DataSupplyRequest Sender="AVMS A" TimeStamp="2001-08-08T05:10:00">
</DataSupplyRequest>
```

In response, the fetcher receives a *DataSupplyAnswer*, in which there is a *FeederMessage* element for every subscription that has changed (6.2.4.1.2). A *CPIDeviation* element is embedded into the feeder message, which contains status information about the trip:

Definition of *CPIDeviation (ASBFahrplanlage)*:

TimeStamp (Zst):	(attribute) Time stamp of the registration of schedule deviation
ValidUntilTimeStamp (VerfallZst):	(attribute) Specifies the time to which the subscription is valid
CPIID (ASBID):	References the connection area
TripID (FahrtID):	(sub-element) References the feeder trip

StopSeqCount (HstSeqZaehler):	Passage counter for circular trip detection (value increases with the number of passages, not sequential)
LineID (LinienID):	Line code of the feeder
LineText (LinienText):	Line name of the feeder (for the passenger)
DirectionID (RichtungsID):	Direction code of the feeder
DirectionText (RichtungsText):	Name of the feeder direction (for the passenger)
AtCPIPoint (AufASB):	(optional) Flag, which indicates that the vehicle has reached the connection area
ScheduledCPIArrivalTime (AnkunftszeitASBPlan):	Planned arrival time of the feeder at the connection area
ExpectedCPIArrivalTime (AnkunftszeitASBPrognose):	Predicted arrival time of the feeder at the connection area stop, as long as *TripStatus* = “real”. Otherwise identical to *ScheduledCPIArrivalTime.*
TripStatus (FahrtStatus):	Specifies whether actual (up-to-date) data can be supplied for the vehicle (“real”) or not (“schedule”).
TransferPassengers (Umsteigewillige):	(optional) Number of transfer passengers. Negative if the information cannot be supplied.
FeederStopLong (ZubringerHstLang):	(optional) Full name of the feeder stop.
LatestFetcherInfoTime (SpaetesteAbbringerInfo)	(optional) time before the fetcher info shall be transmitted.
StopID (HaltID)	(optional) references the stop position within the connection area
StopPositionText (*HaltepositionsText)*	(optional) Describes new stop position.
TripInfo (FahrtInfo):	(sub-element, optional) Sub-element that supplies additional information on the trip.

The element *LatestFetcherInfoTime* is intended to inform the fetcher system when the feeder system wants to use the information (for passenger information). A later transmission can result in a more reliable information.

Example: Trip-based subscription

In the following example we assume that the planned arrival time of the feeder is 3:58 p.m. (for corresponding subscription example see 6.2.4.1.2). The first schedule deviation telegram is sent when the 30-minute preview time is reached. There is no possibility of a prediction at the time of sending the schedule status (3:28 p.m.), as the vehicle is not yet on route (*TripStatus* = “schedule”, *Prediction = “planed”*).

```
<DataSupplyAnswer>
      <Acknowledge TimeStamp="2001-08-08T15:28:00" Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <FeederMessage SubscriptionID="25">
      <CPIDeviation TimeStamp="2001-08-08T05:10:05">
            < ValidUntilTimeStamp=”2001-08-08T16:00:00”>
            <CPIID>12345</CPIID>
            <TripID>
```

```
                  <TripName>7748</TripName>
                  <DayType>2001-08-08</DayType>
            </TripID>
            <StopSeqCount>1</StopSeqCount>
            <ScheduledCPIArrivalTime>
                  2001-08-08T15:58:00
            </ScheduledCPIArrivalTime>
            </ScheduledCPIArrivalTime>
            <ExpectedCPIArrivalTime>
                  2001-08-08T15:58:00
            </ExpectedCPIArrivalTime>
            <StopID>B3</StopID>
            <StopPositionText>Bussteig B</StopPositionText>
            <TripStatus>Schedule</TripStatus>
            <AtCPIPoint>false</AtCPIPoint>
            <LineID>10</LineID>
            <LineText>X10</LineText>
            <DirectionID>Zoo</DirectionID>
            <DirectionText>Zoological Garden</DirectionText>
      </CPIDeviation>
      </FeederMessage>
</DataSupplyAnswer>
```

If the trip has commenced and the schedule deviation exceeds 2 minutes (desired hysteresis in *CPISubscription*), this situation is signalled by the feeder (*DataReadyRequest*). The request occurs again via a *DataSupplyRequest* triggered by the fetcher. A possible response could look as follows:

```
<DataSupplyAnswer>
      <Acknowledge TimeStamp="2001-08-08T15:40:00" Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <FeederMessage SubscriptionID="25">
      <CPIDeviation TimeStamp="2001-08-08T15:39:00">
            < ValidUntilTimeStamp=”2001-08-08T16:00:00”>
                  <TripID>
                        <TripName>7748</TripName>
                        <DayType>2001-08-08</DayType>
            </TripID>
                  <StopSeqCount>1</StopSeqCount>
            <ScheduledCPIArrivalTime>
                  2001-08-08T15:58:00
            </ScheduledCPIArrivalTime>
            <ExpectedCPIArrivalTime>
                  2001-08-08T16:00:00
            </ExpectedCPIArrivalTime>
            <TripStatus>Real</TripStatus>
            <AtCPIPoint>false</AtCPIPoint>
            <LineID>10</LineID>
            <LineText>X10</LineText>
            <DirectionID>Zoo</DirectionID>
            <DirectionText>Zoological Garden</DirectionText>
      </CPIDeviation>
      </FeederMessage>
</DataSupplyAnswer>
```

This procedure is repeated until the vehicle has reached the stop (last message with *AtCPI-Point* = true), or the fetcher deletes the subscription.

Implementation hint:

When *ValidUntilTimeStamp (VerfallZst*) is reached, the data have to be deleted from the connection protection, even if this is before *ExpectedCPIArrivalTime* (*AnkunftszeitASBProg-nose).*

Example: Time-based subscription

In the following example the feeder responds to the *DataSupplyRequest* with the schedule deviations of two vehicles (for the corresponding subscription example see 6.2.4.2.2):

```
<DataSupplyAnswer>
      <Acknowledge TimeStamp="2001-08-08T15:45:30" Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <FeederMessage SubscriptionID="25">
      <CPIDeviation TimeStamp="2001-08-08T15:45:10">
            <ValidUntilTimeStamp=”2001-08-08T16:00:00”>
            <CPIID>12345</CPIID>
                  <TripID>
                        <TripName>7748</TripName>
                        <DayType>2001-08-08</DayType>
            </TripID>
            <StopSeqCount>1</StopSeqCount>
            <ScheduledCPIArrivalTime>
                  2001-08-08T15:48:00
            </ScheduledCPIArrivalTime>
            <ExpectedCPIArrivalTime>
                  2001-08-08T15:53:00
            </ExpectedCPIArrivalTime>
            <StopID>B3</StopID>
            <StopPositionText>Bussteig 3</StopPositionText>
            <TripStatus>Real</TripStatus>
            <AtCPIPoint>false</AtCPIPoint>
            <LineID>2</LineID>
            <LineText>2</LineText>
            <DirectionID>Station</DirectionID>
            <DirectionText>Station</DirectionText>
      </CPIDeviation>
      <CPIDeviation TimeStamp="2001-08-08T15:45:10">
            <CPIID>12345</CPIID>
                  <TripID>
                        <TripName>6611</TripName>
                        <DayType>2001-08-08</DayType>
            </TripID>
                  <StopSeqCount>1</StopSeqCount>
            <ScheduledCPIArrivalTime>
                  2001-08-08T16:05:00
            </ScheduledCPIArrivalTime>
            <ExpectedCPIArrivalTime>
                  2001-08-08T16:06:00
            </ExpectedCPIArrivalTime>
            <TripStatus>Real</TripStatus>
            <AtCPIPoint>false</AtCPIPoint>
            <LineID>2</LineID>
            <LineText>2</LineText>
            <DirectionID>Station</DirectionID>
            <DirectionText>Station</DirectionText>
      </CPIDeviation>
      </FeederMessage>
</DataSupplyAnswer>
```

Trip 7748 is an example of a trip that only falls within the time window due to its severe delay.

Change messages are issued in the same way as in trip-based connection protection. The reader should note that once trips have fallen into the time window, they stay in the window for the entire duration of the subscription and with that continue to be updated and transmitted. The number of reported schedule deviations can therefore increase throughout the life of the subscription (3.1.9).

6.2.4.3.2 Feeder Loss (CPITripDelete)

There are several possible reasons for the loss of a feeder:

→ Loss/cancellation of the feeder vehicle

→ Diversion of the feeder line/direction

→ Disruption of a section of line

→ Early turn back

Loss of a feeder denotes the failure to stop at the connection stop or a delayed arrival at the connection stop. This can lead to the fetcher vehicle abandoning the connection.

The situation is signalled in a similar way to schedule deviations (6.2.4.2.2, 6.2.4.3.1). Occurrence of one of the above-mentioned events is interpreted as a change. The associated message is *CPITripDelete*, which is a sub-element of *FeederMessage*.

Definition of *CPITripDelete* (*ASBFahrtLoeschen)*:

TimeStamp (Zst):	(attribute) Time stamp of the dispatch action
CPIID (ASBID):	References the connection area
TripID (FahrtID):	(sub-element) References the missing feeder trip
StopSeqCount (HstSeqZaehler):	Passage counter of the feeder for detection of circular trips, value increases with the number of passages
LineID (LinienID):	References the feeder line
LineText (LinienText):	Name of the feeder line (for the passenger)
DirectionID (RichtungsID):	Code for feeder direction
DirectionTex (RichtungsText)t:	Name of the feeder direction (for the passenger)
Reason (Ursache):	(optional) Description of the reason for the loss of the trip

The textual descriptions (line, direction) are for informative purposes only (dispatcher), as the trip is uniquely referenced via the *TripID*.

The following example shows a possible message, reported by a feeder following the absence of trip 6611, on line 12, in connection area 12345 at 3:55 p.m.:

```
<DataSupplyAnswer>
      <Acknowledge TimeStamp="2001-08-08T15:56:00" Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <FeederMessage SubscriptionID="25">
      <CPITripDelete TimeStamp="2001-08-08T15:55:00">
            <CPIID>12345</CPIID>
            <TripID>
                  <TripName>6612</TripName>
                  <DayType>2001-08-08</DayType>
            </TripID>
            <StopSeqCount>1</StopSeqCount>
            <LineID>12</LineID>
            <DirectionID>TRPK</DirectionID>
            <LineText>12</LineText>
            <DirectionText>Park</DirectionText>
            <Reason>Vehicle breakdown</Reason>
      </CPITripDelete>
      </FeederMessage>
 </DataSupplyAnswer>
```

It should be noted that the trip itself must not be deleted from the subscription on the basis of the failure message from the feeder. Following a fetcher reset after the loss of data (*DataSupplyRequest*), the failure message must be initially sent again and in place of a *CPIDeviation*.

6.2.4.4 Messages from the Fetcher Vehicles (*FetcherMessage*)

In addition to the messages that can be transmitted from the feeder to the fetcher, there are other messages that are transmitted in the opposite direction, from the fetcher to the feeder. The fetcherl messages are provided with the codes of one or more feeder trips, which enables corresponding notification.

These messages do not require specific subscriptions – they represent the reverse channel of an existing subscription.

The following reverse channel messages are provided:

→ Failure of the fetcher

→ Change to the dispatch status of the fetcher (waiting time)

→ Change of arrival location in the connection area

Definition of *FetcherMessage (Abbringernachricht)*:

SubscriptionID (AboID):	(attribute) Unique reference for the subscription within the service
StopPositionChange (HaltepositionsAenderung):	(sub-element, alternative, multiple) Reports a change to the departure location of the fetcher in the connection area

WaitUntil (WartetBis):	(sub-element, alternative, multiple) Reports a change to the departure time of the fetcher
FetcherTripDelete (AbbringerFahrtLoeschen):	(sub-element, alternative,multiple) Reports the failure of a fetcher trip in the connection area

The fetcher signals the change of the process data associated with the subscription via a *DataReadyRequest*. They are then retrieved via a *DataSupplyRequest by the feeder*. The feeder then deliveres one or more of the above described *FetcherMessages.*

6.2.4.4.1 Change to the Stopping Position (*StopPositionChange*)

If the fetcher changes its planned stopping position within the connection area, this can be relevant to the passenger. This function is used to provide the passenger with information relating to the new connection point.

It is based on an existing subscription (*CPISubscription*), which retrieves feeder data. On the basis of the *TripIDExt* it is possible to send reverse messages relating to one or more feeder trips using the TripID of the fetcher as reference. Thus it is possible, to show the new stop position of the fetcher on the display facilities in the feeder vehicles.

Definition of *StopPositionChange (HaltepositionsAenderung):*

TimeStamp (Zst):	(attribute) Time stamp of the dispatch action
CPIID (ASBID):	References the connection area
FetcherInfo (AbbringerInfo):	(sub-element) Uniquely references the fetcher
TripIDEx (FahrtIDExt)t:	(sub-element, multiple) References the feeder trips

6.2.4.4.2 Referencing of Trips (TripIDExt)

The element *TriptIDExt is used to* identifiy the trips via TripID or TripName, DayType or StopSeqCount, in the case that one fetcher message applies to several feeder trips.

Definition of *TripIDEx (FahrtIDExt)t*:

TripName (Fahrtbezeichner):	Unique reference to the trip depending on the operating day.
DayType (Betriebstag):	Denotes the operational day on which the trip takes place
StopSeqCount (HstSeqZaehler):	Passage counter for circular trip detection. Value increases with the number of passages. Not sequential.

Example:

Let us assume there is a subscription with the ID 25. The fetcher (*TripID* 6612, line 8, direction “Castle Square”) changes its stopping position in the connection area (*CPIID* 12345) to “platform 3”. The message is addressed to feeder trip 5467.

```
<DataSupplyAnswer>
      <Acknowledge TimeStamp="2001-08-08T15:56:00" Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <FetcherMessage SubscriptionID="25">
      <StopPositionChange TimeStamp="2001-08-08T15:55:00">
            <CPIID>12345</CPIID>
            <FetcherInfo>
                  <TripID>
                        <TripName>6612</TripName>
                              <DayType>2001-08-08</DayType>
                  </TripID>
                  <StopSeqCount>1</StopSeqCount>
                  <LineID>8</LineID>
                  <DirectionID>CASSQ</DirectionID>
                  <LineText>8</LineText>
                  <DirectionText>Castle Square</DirectionText>
                  < ScheduledCPIDepartureTime>
                        2001-08-08T16:00:00
                  </ ScheduledCPIDepartureTime>
      </FetcherInfo>
            <TripIDExt>
                  <TripName>5467</TripName>
                  <DayType>2001-08-08</DayType>
                  <StopSeqCount>1</StopSeqCount>
            </TripIDExt>
            <StopPositionText>
                  Platform 3
            </StopPositionText>
      </ StopPositionChange >
      </FetcherMessage>
 </DataSupplyAnswer>
```

6.2.4.4.3 Prolonged Wait (*WaitUntil*)

An important function of connection protection is the ability to hold back the fetcher vehicle to allow passengers to transfer from delayed feeders. This so-called connection dispatch action generally involves several stages. It usually begins as an automatic process, until a certain critical threshold is reached. The decision to prolong the wait is then transferred to the dispatcher.

The result is the prolonged wait of the fetcher in the connection area, ordered by the AVMS.

This event should be communicated to the feeder to inform the passengers of the revised departure time of the fetcher (via the interior displays in the vehicle).

A prolonged wait is reported to the feeder with the *WaitUntil* message within a *FetcherMessage*.

Definition of *WaitUntil (WartetBis)*:

TimeStamp (Zst):	(attribute) Time stamp of the wait decision
CPIID (ASBID):	References the connection area
FetcherInfo (AbbringerInfo):	(sub-element) References the fetcher trip
TripIDExt (FahrtIDExt):	(sub-element, multiple) References the feeder trips
ExpectedCPIDepartureTime (AbfahrtszeitASBPrognose):	Predicted departure time of the fetcher at the connection area stop.
Reliability (Verlaesslichkeit)	(optional) classifies the rliability / probability oft he prognosis (level 1 to 5 with 1 for highest and 5 for lowest vaslue)

Implementation hint:

If a fetcher waits for several feeders an element WaitUntil with several sub-elements *TripIDExt (FahrtIDExt)* is sent. If one of the feeders is that late, that the fetcher can not wait for it, a message *FetcherTripDelete*(*AbbringerFahrtLoeschen)* with its *TripIDExt (FahrtIDExt)* is sent.

If an AVMS works partially without scheduled data; it may be helpful to send also information [*WaitUntil (WartetBis)* and *ExpectedCPIDepartureTime (AbfahrtszeitASBPrognose)]* for fetchers which are in time and for which there is no need for a prolonged waiting time.

The same applies if a connection can be held because both vehicles are late.

Example: Fetcher 6612 of line 8, direction “Castle Square” reports a new departure time of 4:02 p.m. at connection area 12345 to feeder 54676.

```
<DataSupplyAnswer>
      <Acknowledge TimeStamp="2001-08-08T15:56:00" Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <FetcherMessage SubscriptionID="25">
      <WaitUntil TimeStamp="2001-08-08T15:55:00">
            <CPIID>12345</CPIID>
            <FetcherInfo>
                  <TripID>
                        <TripName>6612</TripName>
                        <DayType>2001-08-08</DayType>
                  </TripID>
                  <StopSeqCount>1</StopSeqCount>
                  <LineID>8</LineID>
                  <DirectionID>CASSQ</DirectionID>
                  <LineText>8</LineText>
                  <DirectionText>Castle Square</DirectionText>
                  <ScheduledCPIDepartureTime>
                        2001-0808T16:00:00
                  </ScheduledCPIDepartureTime>
            <StopPositionText>Bussteig 3</StopPositionText>
            </FetcherInfo>
            <TripIDExt>
                  <TripName>5467</TripName>
                  <DayType>2001-08-08</DayType>
                  <StopSeqCount>1</StopSeqCount>
            </TripIDExt>
      <ExpectedCPIDepartureTime>
                  2001-08-08T16:02:00
            </ExpectedCPIDepartureTime>
      </WaitUntil>
      </FetcherMessage>
</DataSupplyAnswer>
```

6.2.4.4.4 Loss of the Fetcher (FetcherTripDelete)

The failure of a fetcher means the failure of the corresponding connection. The *CPITrip-Delete* (6.2.4.3.2) message is used to signal this event to the passengers in the feeder vehicles (via the interior displays), this time however it is used in the opposite direction.

The message concerns the entire subscription, i.e. all previously reported feeders.

To stop the continued transmission of feeder deviations, the associated subscription must be deleted on the side of the fetcher.

Implementation hint:

If the fetcher journey changes, ***FetcherTripDelete* (*AbbringerFahrtLoeschen)*** should be sent together with the information about the new fetcher ***WaitUntil (WartetBis)*** *in a* ***Fetcher-Info*** to avoid that the connection breaks accidentally.

Compared with the *FeederMessage*, the semantics of the elements in *FetcherTripDelete*are slightly different:

Definition of *FetcherTripDelete*: (*AbbringerFahrtLoeschen)*

TimeStamp (Zst):	(attribute) Time stamp of the dispatch action
CPIID (ASBID):	References the connection area
FetcherInfo (AbbringerInfo):	(sub-element) References the missing fetcher trip
TripID (FahrtIDExt):	(sub-element) References the feeder trip
Reason (Ursache):	(optional) Description of the reason for failure

The following example illustrates the message for the failure of a fetcher (7712) of line 8 in connection area 12345 sent to feeder “7748”, whose feeder data was being retrieved under SubscriptionID 25.

```
<DataSupplyAnswer>
      <Acknowledge TimeStamp="2001-08-08T15:56:00" Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <FetcherMessage SubscriptionID="25">
      <FetcherTripDeleteTimeStamp="2001-08-08T15:55:00">
            <CPIID>12345</CPIID>
            <FetcherInfo>
                  <TripID>
                        <TripName>7712</TripName>
                        <DayType>2001-08-08</DayType>
                  </TripID>
                  <LineID>8</LineID>
                  <DirectionID>CASSQ</DirectionID>
                  <LineText>8</LineText>
                  <DirectionText>Castle Square</DirectionText>
                  <ScheduledCPIDepartureTime>
                        2001-0808T16:00:00
                  </ScheduledCPIDepartureTime>
            </FetcherInfo>
            <TripIDExt>
                  <TripName>7748</TripName>
                  <DayType>2001-08-08</DayType>
                  <StopSeqCount>1</StopSeqCount>
            </TripIDExt>
                  </TripIDExt>
      </FetcherTripDelete>
      </FetcherMessage>
 </DataSupplyAnswer>
```

6.3 Dynamic Passenger Information (REF-DPI, DPI)

6.3.1 Introduction

The dynamic passenger information service is used for the agreement and transmission of data to permit the operation of common passenger information displays at the stops. The following specification describes the exchange of full text information to supply foreign dynamic passenger information displays. (Optimised data exchange using text codes in place of full texts is also possible with inter-operational agreements). In this context, one of the participant operators must have ownership of the displays (*display owner*) and with that overall control of them.

The operator who wishes to show passenger information on a foreign display is termed the *display user*.

Data exchange within the interface is achieved using full texts (destination, via stations). If due to minimal bandwidth, the displays need to be controlled using codes, these codes must be agreed by all parties and exchanged within the scope of data management. The full texts must be assigned to the corresponding codes on the side of the display owner system (6.3.3).

As some systems only permit the processing of process data, when corresponding planning data already exists, there is also an optional reference data service for the passenger information (REF-DPI). Where necessary, this permits the exchange of location related planned schedules for the passenger information.

6.3.2 Operational Data Supply and Management

Similar to the procedure for connection protection, the first step is an inter-operational agreement of the location codes, here termed *display areas* (*DIS*) (6.1.3). Unique, bilateral codes are agreed, so-called *DISIDs*. On the side of the display owner, these codes are assigned to the individual displays (or groups of displays):

DISID	Foreign operator code	Internal display code
12345	AVMS B	2345
12346	AVMS B	3687
35678	AVMS B	7566

Table 12: Definition of DPI location codes in the display owner system

On the side of the display user, the codes are not related to the individual displays but to the stops. This yields a cross referencing between the foreign stops and the associated displays. When defining the *DISIDs* it is necessary to account for the possible usage within logoff telegrams for quick cleardown at stops (6.3.4)

If only a selection of lines passing a stop are to be shown on the corresponding displays, it is necessary to provide a table that assigns the foreign lines to the displays:

DISID	Foreign operator code	Foreign line	Direction on foreign line
12345	AVMS B	8	Zoo
12346	AVMS B	10	Central Station

Table 13: Definition of lines to be displayed

There must also be an agreement of product types. This is relevant for the use of symbols to represent the different operating areas or modes of transport on the displays.

Differing field lengths on the displays (e.g. for destination or via texts) present a problem. It cannot be assumed that one field length will suffice. Each type of display requires a corresponding supply with full texts. The interface provides a function which allows the user to specify a maximum length (*MaxTextLength*), which defines the optimum length where several are supplied. But even this function is, to a certain extent, incomplete as proportional fonts are often used on displays. A display therefore does not have a common text length per line but dots per line, which can represent texts of varying lengths. Management of the full texts therefore requires intensive consultation and checking of the texts.

6.3.3 DPI Systems with Code Control

DPI systems in which there is only limited bandwidth for data supply via radio, frequently use predefined texts with assigned codes to control the displays in place of full texts (direction, via stations). Communication between the control system and displays is based on the shorter codes.

Full texts are used exclusively within the interface. If code control is also to be used for the foreign vehicles, all predefined texts including those of the display users, must be imported and given codes. These must then be made known to the display system within the scope of data supply. Conversion to the full text is carried out by the display owner system by means of comparing the full text with the text/code table of the foreign operator.

6.3.4 DPI Systems with Autonomous Predictions

The physical message channel between the AVMS and DPI displays often represents a very restricted resource (e.g. narrow band radio network). The reference service for DPI provides information to make better use of this limited resource. The idea is to eliminate redundant information that results from the individual control of each individual display. The procedure is to supply the displays with the schedule information at the start of the operating day. During operation, instead of the predicted departure times to the displays served by a trip, only the delay information is transmitted in broadcast mode. The display filters the relevant data packets from the flow of data and itself calculates the predicted arrival times from the planning data supply and the received real-time information. They are sometimes referred to as “intelligent displays”.

The control of autonomous displays demands trip-specific planned and real-time data. The DPI interface however only provides location-specific planned or real-time data, and is not therefore in a position to operate systems for controlling autonomous displays.

For such DPI systems it is possible to rely on the schedule information interface (VDV Recommendation 454), as this interface is capable of providing trip or line specific data.

6.3.5 Quick Cleardown

The so-called "quick cleardown" function is used to delete represented trip information from the displays as soon as the vehicle departs from the stop. For this, the stop/display must be able to communicate directly with the vehicle. TLP (traffic light priority) radio technology is usually employed.

When the vehicle departs from the stop it sends a radio message with the code of the trip. The display can then delete the corresponding information.

The code (*TripID*) that is used within the interface to identify the trip cannot usually be used: This is an internal AVMS code that is not provided in the vehicle data supply.

In order to be able to implement quick cleardown in spite of this, a so-called *DepartureNoticeID* is added to the *TripID* in all relevant messages. This is known in the vehicles of the display user system and having received this information the display owner system makes it known to the displays within the context of its schedule data supply.

Contrary to the *TripID*, the *DepartureNoticeID* is only unique on the operating day of the display user system. Additional logics may be required in the limiting cases. The situation could arise in which the display is assigned trips with the same *DepartureNoticeID* on the basis of the current time and the planned departure time.

In addition to the *DepartureNoticeID*, which identifies the vehicle or trip, a location reference is also often needed defining the display from which the trip is to be deleted. This distinction is necessary when there are several displays within the radio range of the vehicle. The *DISID* should be used for this purpose. If this cannot be used due to the specifications of existing systems (type, format, size), the display owner display reference used in the deletion process must be stored in addition to the *DISID* in the operational data supply of the display user (Table 14). This must also be loaded into the onboard computer in the scope of supplying the vehicle data supply. When defining the *DISID* it is recommended to support the used physical data format of the quick cleardown. Otherwise, additional operational data management is required, as described.

Foreign operator code	DISID	Foreign display reference
AVMS B	12345	XK3
AVMS B	45678	ZZ4

Table 14: Foreign display references for quick cleardown

6.3.6 Trainsets / Run Vehicles / Splitting or Combining Trips

The term trainset describes the combination of several individual trips. These combinations can be operated along the route connected together or separately (splitting or combining trips), but they can also serve the entire route jointly.

From the point of view of the passenger, these combinations should be displayed in the DPI as a single trip or as individual trips, depending on the given situation. Furthermore, in some systems the position within the combination is also relevant, in order to supply the destination displays in the various stop areas with the respective trip information.

In order to allow this, reference to an existing trainset is included in the DPI messages (*TrainsetID*). This allows the individual parts of a train assembly to be identified. Other optional elements determine the total number and position of the individual trips (*NumOfTrips, Position*).

Specification of trainset data is optional. Contrary to the standard behaviour for optional elements, existing trainset data indicates that a trip at a stopping point is part of a trainset.

The transmission of trainset data starts at the stopping point at which the vehicle is coupled together and ends at the stopping point at which the vehicle is decoupled, or at which the trip leaves the trainset. Trips can switch trainsets and several trainsets can be combined to form a new one.

6.3.7 Reference Data Service (REF-DPI)

6.3.7.1 Data Exchange

The exchange of planning data for the dynamic passenger information system is used to create reference data within the data consuming system (display owner system). Not all systems require such reference data in order to be able to process the associated process data. The exchange of reference data can then be omitted for the dynamic passenger information system.

The REF-DPI reference data service is largely identical to the REF-CP service. In both cases it involves an exchange of departure times of vehicles at a foreign stop/display. The differences lie in the nomenclature (display area / connection area), as well as in the format of the data being exchanged.

6.3.7.1.1 Updating

As in the REF-CP service, the data is updated in the case of additional trips. Missing or modified trips are not communicated or registered as additional trips.

6.3.7.2 Requesting DPI Reference Data (*DISRefSubscription*)

A *SubscriptionRequest* with one or more embedded *DISRefSubscription* elements is used to request the schedule data.

The *DISRefSubscription* element specifies the display area, optional filters as well as the request time period.

Definition of *DISRefSubscription (AboAZBRef)*:

SubscriptionID (*AboID*):	(attribute) References the subscription for DPI planning data created by the request. The SubscriptionID is assigned by the display owner system.
ValidUntilTimeStamp (*VerfallZst*):	(attrib.) Specifies the time to which the subscription is valid
DISID (AZBID):	References the display area
LineID (LinienID):	(optional) Filter for foreign line whose data is to be supplied
DirectionID (RichtungsID):	(optional) Filter for foreign direction whose data is to be supplied
EarliestDepartureTime: (FruehesteAbfahrtsszeit)	Defines the start of the time period for which data is to be supplied. The reference is the departure time of the foreign vehicle at the display area.
LatestDepartureTime: (SpaetesteAbfahrtsszeit)	Defines the end of the time period for which data is to be supplied. The reference is the departure time of the foreign vehicle at the display area.

LineID and *DirectionID* are independently optional. It is therefore possible to specify a request that has a direction filter but no line filter.

The *EarliestDepartureTime* element should not be set to a time before the data horizon, and *LatestDepartureTime* should not be after the end of the horizon. The *ValidUntilTimeStamp* attribute should be the same time as or later than *LatestDepartureTime.*

The following example describes a reference data request (“AVMS A” is the display owner system) for display area “12345”. Data from line 10 in direction “Zoo” is to be supplied. The display owner AVMS, which created the request, has a data horizon from 5:00 a.m. to 11:00 p.m. on 8.8.2001. The foreign AVMS only has a data horizon to 10:00 p.m.

```
<SubscriptionRequest Sender="AVMS A" TimeStamp="2001-08-08T05:00:00">
      <DISRefSubscription SubscriptionID="25" ValidUntilTimeStamp="2001-08-
09T00:00:00">
      <DISID>12345</DISID>
      <LineID>10</LineID>
      <DirectionID>Zoo</DirectionID>
      <EarliestDepartureTime>
            2001-08-08T05:00:00
      </EarliestDepartureTime>
      <LatestDepartureTime>
            2001-08-08T23:00:00
      </LatestDepartureTime>
      </DISRefSubscription>
</SubscriptionRequest>
```

The feeder responds with an *Acknowledge* within *SubscriptionReply*. The restricted data horizon is announced in *DataValidUntil*:

```
<SubscriptionReply>
      <Acknowledge TimeStamp="2001-08-08T05:00:05" Result="ok"
            ErrorNumber="0">
            <DataValidUntil>2001-08-08T22:00:00</DataValidUntil>
      </Acknowledge>
</SubscriptionReply>
```

6.3.7.3 Transferring REF-DPI Data (*DISSchedule*)

Once the reference data subscription is set up (6.3.7.2), the display user system establishes the arrival times of the individual vehicles at the display area and signals this with an initial *DataReadyRequest* message.

After reception and confirmation of the ready message the display owner system polls the data with a *DataSupplyRequest*.

The display user system responds with the requested data within *DISSchedule* elements. The *DISSchedule* in turn is a sub-element of the so-called *DISMessage (6.3.8.3)* and corresponds to a concrete arrival in a display area. A *DISMessage* is directly assigned to a subscription.

The arrivals table is therefore formed by a list of several *DISSchedule* elements.

Definition of *DISSchedule (AZBFahrplan)*:

TimeStamp (Zst):	(attribute) Time of the schedule data change.
DISID (AZBID):	Display area reference.
StopSeqCount (HstSeqZaehler):	Passage counter for detecting circular trips, value increases with the number of passages, not sequential
TripID (FahrtID):	(sub-element) Foreign trip reference
LineID (LinienID):	Line code of the foreign trip
LineText (LinienText):	Line name of foreign trip (for the passenger)
DirectionID (RichtungsID):	Direction code of the foreign trip
DirectionText (RichtungsText):	Name of the direction of the foreign trip (for the passenger)
ScheduledDISArrivalTime (AnkunftszeitAZBPlan):	optional Scheduled arrival time of the foreign trip in the display area
ScheduledDISDepartureTime (AbfahrtszeitAZBPlan):	optional Schedule departure time of the foreign trip in the display area
TripInfo (FahrtInfo):	(optional, sub-element) Additional information on the foreign trip

In order to correctly display starting and ending journeys, the elements *ScheduledDISArrivalTime (AnkunftszeitAZBPlan) and ScheduledDISDepartureTime (AbfahrtszeitAZBPlan)* are optional. But at least one element has to be provided.

6.3.8 Process Data Service (DPI)

6.3.8.1 Data Exchange

The exchange of process data for the DPI provides the display owner system with information about the trips of a foreign operator to be shown on the displays. The provision of this data occurs within a given preview time window. The amount of trip information supplied can also be restricted to the amount of information that is to appear on the displays.

The procedure of data exchange is illustrated in Figure 2.

The display owner system polls the data for the trips of specific lines and directions that are to appear on the displays (*DISSubscription*). It is possible to specify the so-called preview time, which defines at what time before the vehicles arrive at the display transmission of schedule status predictions is to start. Once the preview time has been reached and after every change, the schedule statuses of the corresponding vehicles are transmitted (*DISDeviation*). The detection of a change in schedule status can be controlled by specifying a *Hysteresis*. Arrival at the display area is also considered to be a change (DataReadyRequest2). The transmission of data for a subscription occurs when the *PreviewTime* is reached.

The subscription also includes a definition of the maximum number of vehicles. This allows the volume of data to be restricted to the number of trips that the display is able to represent.

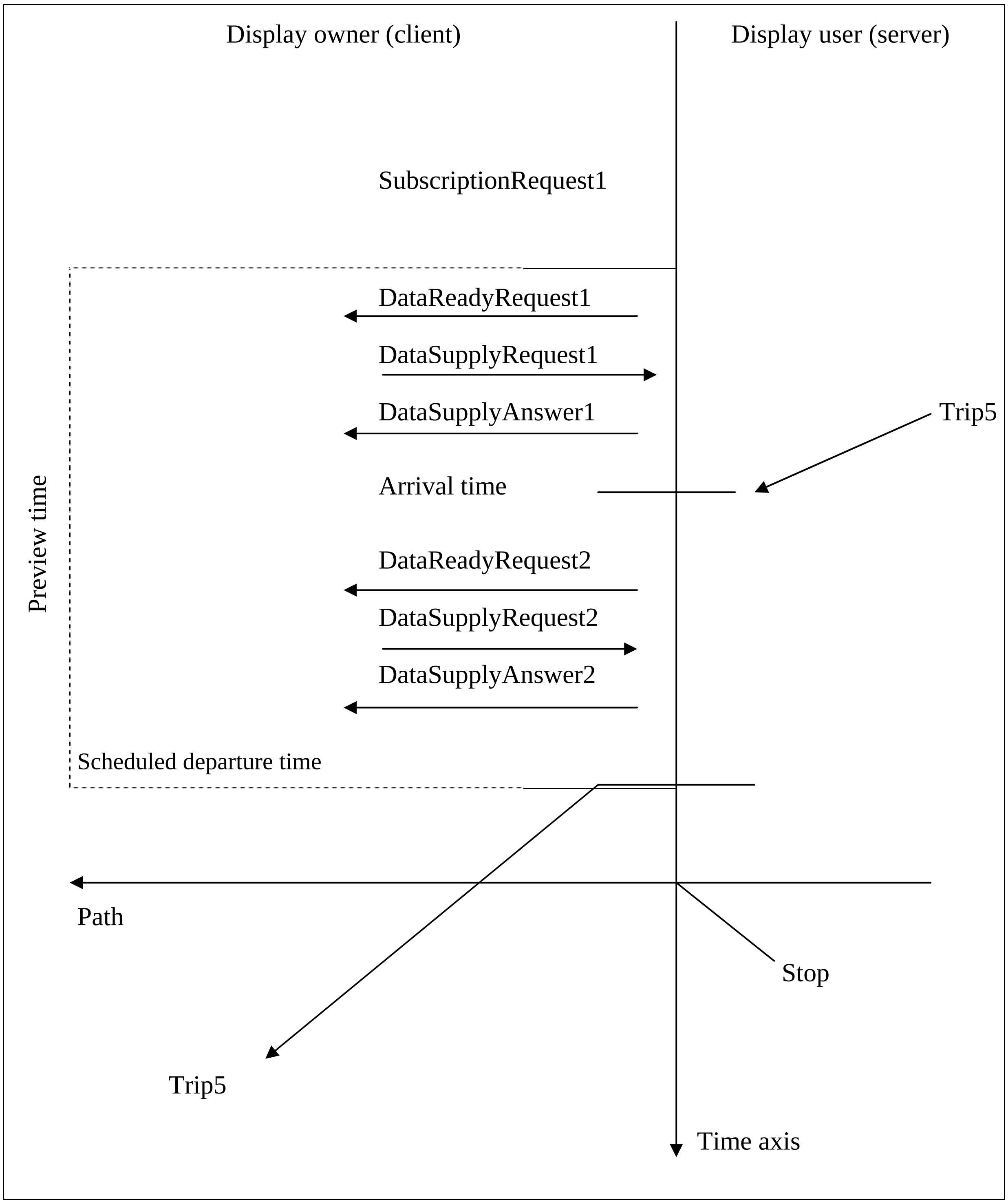

Figure 2: Data exchange for DPI

Polling occurs as soon as possible (at the start of the operational day) and ends at the end of the day. As with the reference data service for connection protection, it must be assumed that it is not possible to cover the entire time period. For this reason, there is a feedback message in *Acknowledge* stating the end of the time horizon of the system supplying the data.

6.3.8.2 Requesting DPI Data (*DISSubscription*)

The polling of DPI data is initiated by the display owner system. It sets up a *SubscriptionRequest*, which must contain one or more *DISSubscription* sub-elements.

Definition of *DISSubscription (AboAZB)*:

SubscriptionID(AboID) :	(attribute) The SubscriptionID references the subscription of display data set up by the request. The SubscriptionID is assigned by the display owner system.
ValidUntilTimeStamp (*VerfallZst*) :	(attribute) Specifies the time to which the subscription is to be valid
DISID (AZBID):	References the display area
LineID (LinienID):	(optional) Reference of the display user line for which data is to be supplied
DirectionID (RichtungsID):	(optional) Indicates the direction of the display user system for which data is to be supplied
PreviewTime (Vorschauzeit):	Specifies the time before the arrival of a trip to be shown on the display at which the transmission of schedule status predictions is to start (in minutes).
MaxNumOfTrips (MaxAnzahlFahrten):	Defines the maximum number of schedule statuses to be reported in *DISDeviation*. The first n vehicles are supplied according to the order they arrive in the display area
Hysteresis (Hysterese):	Required change value in seconds after which schedule deviation update is to be communicated to the display owner system.
MaxTextLength (MaxTextLaenge):	(optional) Required maximum text lengths of the destination and via texts. This value is only a recommendation. Longer texts may also be supplied.

LineID and *DirectionID* are independently optional. It is also possible therefore to set up a request without any filters. *MaxNumOfTrips* serves to optimise the data traffic. The display owner system can restrict the number of trips to be provided in a *DISDeviation*. This means it is possible to restrict the number of messages to the number that are to be represented on the display. The system supplies the trips that are next to reach the display area. As in the case of time-based connection protection, schedule status predictions once sent for particular trips must continue to be sent. Dispatch actions can result in more schedule status predictions being sent than the number defined in *MaxNumOfTrips*. The following table clarifies the procedure:

Table 15 shows a departure table (of the display user system) of the trips serving the display area. *MaxNumOfTrips* in the subscription has been assigned the value 3. After reaching the preview time, only the trips highlighted in grey should be reported.

TripID	Predicted departure time
123	1:00 p.m.
124	1:10 p.m.
125	1:20 p.m.
126	1:30 p.m.
127	1:40 p.m.
128	1:50 p.m.

Table 15: Departure table restricted by MaxNumOfTrips

If a reinforcement trip 566 is now deployed with predicted departure time of 1:05 p.m., this pushes trip 125 out of third place. Despite this, trip 125 continues to be reported (see Table 16), as it has already been communicated once. Within a *DISMessage* there may now be a maximum of four *DISDeviations*.

TripID	Predicted departure time
123	1:00 p.m.
566	1:05 p.m.
124	1:10 p.m.
125	1:20 p.m.
126	1:30 p.m.
127	1:40 p.m.
128	1:50 p.m.

Table 16: Exception to the restriction as a result of a reinforcement trip

Implementation hint:

If there are fixed areas on the display for lines / directions or if the next trip is to be displayed per line / direction, then the subscriptions should also be set up specific to line / direction.

If the value for hysteresis has been exceeded for only a part of the journeys transmitted, only these data have to be retransmitted.

In the following example, data should be subscribed for a display (owner is AVMS A) with *DISID* “12345”. Data is to be transmitted up to 11:00 p.m. with a *PreviewTime* of 30 minutes. The text fields should have a maximum length of 30 characters (*MaxTextLength*). The display user should report a maximum of 3 trips of line 8 in all directions:

```
<SubscriptionRequest Sender="AVMS A" TimeStamp="2001-08-08T05:00:00">
      <DISSubscription SubscriptionID="25" ValidUntilTimeStamp="2001-08-
08T23:00:00">
            <DISID>12345</DISID>
            <LineID>8</LineID>
            <PreviewTime>30</PreviewTime>
            <MaxNumOfTrips>3<MaxNumOfTrips>
            <Hysteresis>120</Hysteresis>
            <MaxTextLength>30</MaxTextLength>
      </DISSubscription>
</SubscriptionRequest>
```

The display user confirms reception of the subscription and reports back immediately that it can only supply data to 10:00 p.m.:

```
<SubscriptionReply>
      <Acknowledge TimeStamp="2001-08-08T05:00:05" Result="ok"
            ErrorNumber="0">
            <DataValidUntil>2001-08-08T22:00:00</DataValidUntil>
      </Acknowledge>
</SubscriptionReply>
```

6.3.8.3 Messages from the Display User System (*DISMessage*)

In the case of updates, the display user responds with a so-called *DISMessage* upon reaching the preview time. With a reference to a subscription, this element forms the frame for the results of dispatch actions or schedule statuses.

The following messages can be reported within the *DISMessage*:

Reference Data Service:

→ (*DISSchedule*)

Process Data Service:

→ Schedule status predictions (*DISDeviation*)

→ Cancellation of a trip, departure on display (*DISTripDelete*)

→ Creation and deletion of a special line text (*DISLineSpecialText* / *DISLineSpecialTextDelete*)

Definition *of DISMessage* **:**	
SubscriptionID (AboID):	(attribute)The SubscriptionID references the subscription of feeder data created by the request. The SubscriptionID is specified by the fetcher system.
DISSchedule (AZBFahrplan):	(sub-element, optional,multiple) Schedule-Informatione for a trip.
DISDeviation (AZBFahrplanlage):	(sub-element, optional,multiple) Schedule status predictions for a trip.
DISTripDelete (AZBFahrtLoeschen):	(sub-element, optional,multiple) Cancellation of a trip, departure on display
DISLineSpecialText (AZBLinienSpezialtext):	(sub-element, optional,multiple) Creation of a special line text
DISLineSpecialTextDelet:-(AZBLinienSpezialtextLoeschen)	(sub-element, optional,multiple) Deletion of a special line text

Each *DISMessage* can contain as many sub-elements as desired.

6.3.8.3.1 Transferring Predictions (*DISDeviation*)

Once the preview time has been reached, transmission of the actual prediction data starts.

In the above example, this is 30 minutes before the vehicle (line 8) reaches the display area. The display user creates an initial schedule status prediction and signals that it is ready to be retrieved (*DataReadyRequest*). The display owner confirms this (*DataReadyAnswer* / *Acknowledge*) and polls the subscription data (DataSupplyRequest). The display user responds with a corresponding *DISMessage*.

This groups together the schedule status predictions (*DISDeviation*) of all approaching trips associated with the subscription.

Definition of *DISDeviation (AZBFahrplanlage)*:	
TimeStamp (Zst) :	(attribute) The time stamp records the time of data creation
ValidUntilTimeStamp (*VerfallZst*) :	(attribute) Specifies the time to which the message is to be valid
TripID (FahrtID):	(sub-element) Uniquely identifies the trip to be displayed
DISID (AZBID):	Identifies the display area
StopSeqCount (HstSeqZaehler):	Passage counter for circular trips, value increases with number of passages
Trainset (Traktion):	(sub-element, optional) Contains trainset data relating to the trip
VehicleNumber (BetrieblicheFahrzeugnummer)	(optional, multiple) Company specific Reference oft the individual vehicle driving in a trainset.

LineID (LinienID):	Reference of the display user line for which data is to be supplied
DirectionID (RichtungsID):	Identifies the direction associated with the trip
LineText (LinienText):	Name of the foreign line (for the passenger)
DirectionText (RichtungsText):	Name of the foreign direction (for the passenger)
DestinationStop (ZielHst):	Short name of the destination stop
ViaStop1 (ViaHst1Lang):	(optional) Full name of via stop 1.
ViaStop2 (ViaHst2Lang):	(optional) Full name of via stop 2.
ViaStop3 (ViaHst3Lang):	(optional) Full name of via stop 3.
DepartureNoticeID (AbmeldeID):	(optional) Internal trip number for quick cleardown
AtDISPoint (AufAZB):	(optional) Specifies that the vehicle is at the display area stop
ScheduledDISArrivalTime (AnkunftszeitAZBPlan):	(optional) Planned arrival time at the display area stop
ExpectedDISArrivalTime (AnkunftszeitAZBPrognose):	(optional) Predicted arrival time at the display area stop, if Trip-Status = “real”, otherwise scheduled time
ScheduledDISDepartureTime (AbfahrtszeitAZBPlan):	(optional) Scheduled departure time from the display area stop
ExpectedDISDepartureTime (AbfahrtszeitAZBPrognose):	(optional) Predicted departure time from the display area stop if TripStatus = “real”, otherwise scheduled time
AimedDISDepartureTime (AbfahrtszeitAZB-Disposition)	(optional) aimed departure time at the stop of the display area, following the updated scheduling of the present moment.
TripStatus (FahrtStatus):	Specifies whether real-time information can be supplied for the vehicle (“real”) or not (“schedule”).
TripSpecialText (Fahrtspezialtext):	(optional) Text that is to be displayed in place of the actual trip data. A special trip text must be deleted if this element is not included.
SpeechOutput	(optional, mutiple) shows if *TripSpecialText (*Fahrtspezialtext) shall be announced additionally via speech output and in which language.
StopID (HaltID)	(optional) references the stop position within the connection area
StopPositionText (HaltepositionsText):	(optional) Textual description of the trip’s arrival location in the display area
QueueIndicator (Stauindikator):	(optional) Specifies whether the vehicle is in a traffic jam (true) or not (false).
TripInfo (FahrtInfo):	(optional, sub-element) Extra information on the current trip

Implementation hints:

When *ValidUntilTimeStamp* (*VerfallZst) is exceeded*, the respective journey has to be deleted on the display, even if *ExpectedDISDepartureTime (AbfahrtszeitAZBPrognose)* or *ExpectedDISArrivalTime (AnkunftszeitAZBPrognose)* has not been reached.

In order to correctly display starting and ending journeys, the elements *ScheduledDISArrivalTime (AnkunftszeitAZBPlan) and ScheduledDISDepartureTime (AbfahrtszeitAZBPlan)* are optional. But at least one element has to be provided.

Foreign control centers have to be provided with all the information about all productive journeys to enable the display of arrivals and departures

- For in between stops *DISDeviation (AZBFahrplanlage)* holds all arrival and departure times.
- Ending journeys or journeys with non productive departures have no departure times. Arrival times for these journeys are transmitted if the arrival is productive and their direction is in line with the direction subscribed.

- Starting journeys and journeys with non-productive arrivals have no arrival times. Departure times for starting journeys are transmitted if the departure is productive and the direction is in line with the direction subscribed.

6.3.8.3.2 Definition of Trainset (Traktion)

Definition of *Trainset (Traktion)*:

TrainsetID (TraktionsID):	Uniquely identifies the trainset
NumOfTrips (AnzahlFahrten):	(optional) Number of individual trips in the trainset. If not specified, then there is no available information on the number of trips within the trainset.
Position (Position):	(optional) Position of the trip within the assembly (1 = first position in the direction of travel). If not specified, it is not possible to provide any trainset position information.

In the following example the display user system responds to the above subscription (25) with the schedule status of one vehicle (2367). The trip is delayed by one minute and occupies the first position in a trainset with two other trips in one assembly.

If there are several trips within the preview time of a subscription, these must all be reported in separate *DISDeviation* elements.

```
<DataSupplyAnswer>
      <Acknowledge TimeStamp="2001-08-08T07:30:15" Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <DISMessage SubscriptionID="25">
            <DISDeviation TimeStamp="2001-08-08T07:30:00>
                  <TripID>2367</TripID>
                  <StopSeqCount>1</StopSeqCount>
                  <DISID>12345</DISID>
                  <Trainset>
                        <TrainsetID>234234</TrainsetID>
                        <NumOfTrips>3</NumOfTrips>
                        <Position>1</Position>
                  </Trainset>
                  <LineID>8</LineID>
                  <LineText>8</LineText>
                  <DirectionID>CST</DirectionID>
                  <DirectionText>Central Station</DirectionText>
                  <DepartureNoticeID>3426</ DepartureNoticeID >
                  <AtDISPoint>false</AtDISPoint>
                  <DestinationStop>Central Station</DestinationStop>
                  <ViaStop1>Market Place</ViaStop1>
                  <ViaStop2>Zoo</ViaStop2>
                  <ViaStop3></DestinationStop3>
                  <ScheduledDISArrivalTime>
                        2001-08-08T08:00:00
                  </ScheduledDISArrivalTime>
                  <ExpectedDISArrivalTime>
                        2001-08-08T08:01:00
                  </ExpectedDISArrivalTime>
                  <ScheduledDISDepartureTime>
                        2001-08-08T08:01:00
                  </ScheduledDISDepartureTime>
                  <ExpectedDISDepartureTime>
                        2001-08-08T08:02:00
                  </ExpectedDISDepartureTime>
            <TripStatus>Real</TripStatus>
                  <StopPositionText>Platform 3</StopPositionText>
                  <TripInfo>
                    ...
                  </TripInfo>
            </DISDeviation>
            ... another two deviations
      </DISMessage>
</DataSupplyAnswer>
```

This process is repeated as soon as there is any change in the subscription.

No more updates are sent once the display area has been reached. The subscription must be deleted.

6.3.8.3.3 Transmitting Special Line Texts (*DISLineSpecialText*)

The special line text is used to inform the passengers about line-specific events (e.g. blocked road).

Displayed special line texts must either be specifically deleted by the display user system (6.3.8.3.4), or they are removed by the display owner system when the subscription expires.

Special line texts are reported by the display user system by means of a *DISLineSpecialText* element within *DISMessage*.

Definition of *DISLineSpecialText (AZBLinienSpezialtext)*:

TimeStamp (Zst):	(attribute) The time stamp defines the time at which the data was recorded
ValidUntilTimeStamp (*VerfallZst*) :	(attribute) Specifies the time to which the message is to be valid
DISID (AZBID):	Identifies the display area
LineID (LinienID):	Reference to the line of the display user system for which a special text is to be displayed
LineText (LinienText):	Name of the line for which the LineSpecialText shall be shown
DirectionID (RichtungsID):	Identifies the direction for displaying the special text
LineSpecialText (Linienspezialtext):	Text to be represented
Priority (Prioritaet):	(optional) Relevance of the Information (value range 1 to 3, 1 highest and 3 lowest priority). Might be used to determine the display format (permanent, blinking, rolling,…).
SpeechOutput (Sprachausgabe)	(optional, mutiple) shows if *TripSpecialText (*Fahrtspezialtext) shall be announced additionally via speech output and in which language.

In the following example, and according to the subscription details above, a text is to be displayed for line 8, direction “Central Station”, informing the passengers of suitability for the disabled:

```
<DataSupplyAnswer>
```

```
      <Acknowledge TimeStamp="2001-08-08T07:30:15" Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <DISMessage SubscriptionID="25">
            <DISLineSpecialText TimeStamp="2001-08-08T07:30:00>
                  <DISID>12345</DISID>
                  <LineID>8</LineID>
                  <DirectionID> CST </DirectionID>
                  <LineSpecialText> All vehicles on line 8, direction
                  Central Station, are low floor buses offering easy
                  access to the disabled.
                  </LineSpecialText>
            </DISLineSpecialText>
      </DISMessage>
</DataSupplyAnswer>
```

It is possible to send several special line texts within one message (in accordance with the definition of *DISMessage*). The operator should ensure the special line texts do not overlap.

6.3.8.3.4 Deleting Special Line Texts (*DISLineSpecialTextDelete*)

Deletion of a special line text is similar to set-up. The *LineSpecialText* element is omitted:

Definition of *DISLineSpecialTextDelete (AZBLinienSpezialtextLoeschen)*:

TimeStamp (Zst):	(attribute) The time stamp defines the time at which the data was recorded
DISID (AZBID):	Identifies the display area
LineID (LinienID):	Reference to the line of the display user system for which a special text is to be displayed
DirectionID (RichtungsID):	Identifies the direction for displaying the special text

The following example deletes the previously created special text:

```
<DataSupplyAnswer>
      <Acknowledge TimeStamp="2001-08-08T07:30:15" Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <DISMessage SubscriptionID="25">
            <DISLineSpecialTextDelete TimeStamp="2001-08-08T07:30:00>
                  <DISID>12345</DISID>
                  <LineID>8</LineID>
                  <DirectionID>CST</DirectionID>
            </DISLineSpecialTextDelete>
      </DISMessage>
</DataSupplyAnswer>
```

6.3.8.3.5 Trip Failure / Departure (DISTripDelete)

If a trip that has already been registered in a display area is cancelled, or if the trip leaves the display area, the display user system must send a message to delete it. For this purpose, the display user sends one or more *DISTripDelete* elements within *DISMessage*. The display owner system must now remove the trip from the display or show a corresponding message to the passenger.

Definition of *DISTripDelete (AZBFahrtLoeschen):*

TimeStamp (Zst):	(attribute) The time stamp specifies the time at which the data was recorded
DISID (AZBID):	Identifies the display area
TripID (FahrtID):	(sub-element) Identifies the missing trip
StopSeqCount (HstSeqZaehler):	Passage counter for circular trips; value increases with number of passages; not sequential
LineID (LinienID):	Identifies the line associated with the missing trip
DirectionID (RichtungsID):	Identifies the direction associated with the missing trip
LineText (LinienText):	Name of the fetcher line (for the passenger)
DirectionText (RichtungsText):	Name of the fetcher direction (for the passenger)
DepartureNoticeID (AbmeldeID):	(optional) Used for quick cleardown
Reason (Ursache):	(optional) Description of the reason for failure, omitted in the case of a normal trip (departure from DIS).

The following example illustrates failure of trip (6612) at 3:55 p.m. in display area “12345”

```
<DataSupplyAnswer>
      <Acknowledge TimeStamp="2001-08-08T15:56:00" Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <DISMessage SubscriptionID="25">
      <DISTripDelete TimeStamp="2001-08-08T15:55:00">
            <TripID>
                  <TripName>6612</TripName>
                  <DayType>2001-08-08</DayType>
            </TripID>
            <StopSeqCount>1</StopSeqCount>
            <DISID>12345</DISID>
            <LineID>8</LineID>
            <DirectionID>CST</DirectionID>
            <LineText>8</LineText>
            <DirectionText>Central Station</DirectionText>
            <Reason>Engine failure</Reason>
      </DISTripDelete>
      </AZBMessage>
 </DataSupplyAnswer>
```

With the failure message, no more updates are reported for this trip.

6.4 Visualisation of Foreign Vehicles (VIS)

6.4.1 Introduction

The “visualisation of foreign vehicles” service (VIS) is used to exchange trip information for display in a foreign AVMS. The following information is then available in the foreign AVMS:

→ Trip information (ID, internal number, service characteristics)

→ Geographical position (longitude, latitude), optional

→ Current position on the line, optional

This permits the following representations:

→ Vehicle list (table)

→ GIS map representation

→ Line diagram

The representations that can be displayed depend on the type of data that is made available by the data producing system. As the geographical position and current route section are optional, they are also prerequisites for the corresponding representations.

The line diagram also demands the availability of route models for the vehicles to be displayed. This interface does not cover the exchange of this data.

The visualisation service only transmits process data.

6.4.2 Operational Data Supply and Management

Visualisation data must be polled. The corresponding subscription is achieved via so-called visualisation areas, which specify the volume of data to be transmitted.

Similar to the other location designators (6.1.4), the visualisation areas (*VISID*) must be agreed by the participant operations and are subject to separate data management. Visualisation areas can be based on lines, groups of lines or as a number of trips in any desired context.

Visualisation areas may consist of only one line. In this case one VISID corresponds to one Line. But it may also consist of several lines. In this case a unique reference has to be established by further specifications.

A visualisation area may also be defined by a spatial definition, ex. by an enumeration of stop points, thus referring to all lines stopping at these points.

6.4.3 Process Data Service

6.4.3.1 Procedure

The exchange of visualisation data starts with the creation of a subscription by the displaying system. It creates subscriptions related to the desired visualisation areas, for which data is to be supplied.

Once the subscription has been set up, trip information is transmitted by the data producers to the visualisation system throughout its entire validity period. The method and frequency of repetition is a matter for the data producer, but can be specified by the displaying system in the scope of a recommendation. Information is only transmitted for vehicles currently on route.

Modified records only are communicated. Information for individual trips may be left out of successive messages, if there have been no changes to position, delay or other relevant data.

If a vehicle leaves the visualisation area, the data producer sends a corresponding message, which permits the displaying system to remove it from the representations. In semantic terms the departure from the visualisation area depends on its definition, but operationally can be reduced to the following points:

- → Trip has finished.
- → Trip has failed / been cancelled.
- → Trip has left the zone (spatial definition)

In general, the transmitted information should provide as complete and accurate a picture of the operation as possible.

6.4.3.2 Subscription request (***AboVIS***)

The polling of visualisation data is initiated by the displaying system with the transmission of a subscription request. This *SubscriptionRequest* contains one or more *VISSubscription* elements. The *VISSubscription* specifies the visualisation area, the requested time frame as well as the desired update cycle (recommendation).

Definition of *VISSubscription (AboVIS)*:

SubscriptionID (AboID):	(attribute) The SubscriptionID references the subscription of visualisation data created by the request. The SubscriptionID is specified by the displaying system.
ValidUntilTimeStamp (VerfallZst):	(attribute) Specifies the time to which the subscription is valid
VISID (VISID):	Identifies the visualisation area
LineID (LinienID):	(optional) Restricts the volume of data to the specified line
DirectionID (RichtungsID):	(optional) Restricts the volume of data to the specified direction
Cycle (Zyklus):	Time interval in seconds in which new data is to be transmitted

The following example shows the creation of a subscription for the visualisation area 12345 in the time period between 5:00 a.m. and 10:00 p.m. New data is to be transmitted every 120 seconds:

```
<SubscriptionRequest Sender="AVMS A" TimeStamp="2001-08-08T05:00:00">
      <VISSubscription SubscriptionID="25" ValidUntilTimeStamp="2001-08-
08T22:00:00">
      <VISID>12345</VISID>
      <Cycle>120</Cycle>
      </VISSubscription>
</SubscriptionRequest>
```

The data producer confirms reception and set-up of the subscription with an *Acknowledge* within *SubscriptionReply*.

6.4.3.3 Messages of the Visualisation Services (*VISMessage*)

All messages of the data producing system are encapsulated within the so-called *VISMessage*. The *VISMessage* is the frame for the subscription and contains the following sub-elements:

→ Trip information (*VISDeviation*,6.4.3.3.1)

→ Trip quits visualisation area (*VISTripDelete*)

Definition of *VISMessage (VISNachricht)*:

SubscriptionID (AboID):	(attribute) The SubscriptionID references the visualisation data subscription created by the request. The SubscriptionID is specified by the displaying system.
VISDeviation (VISFahrplanlage):	(sub-element, optional, multiple) Information on a trip
VISTripDelete (VISFahrtLoeschen):	(sub-element, optional, multiple) Instructs the receiver to remove the trip from the representations

6.4.3.3.1 Transferring Visualisation Data (VISDeviation)

The data producer signals the readiness of new data with a *DataReadyRequest* message. The displaying system acknowledges this with a *DataReadyAnswer*.

It then requests the data by sending a *DataSupplyRequest*:

```
<DataSupplyRequest Sender="AVMS B" TimeStamp="2001-08-08T05:05:00">
      <AllData>true</AllData>
</DataSupplyRequest>
```

The data producer supplies the trip information in the reply.

Definition of *VISDeviation (VISFahrplanlage)*:

TimeStamp (Zst):	(attribute) The time stamp registers the time of recording
ValidUntilTimeStamp (VerfallZst):	(attribute) Specifies the time at which the trip is to be removed from the visualisation, in the absence of further data
VISID (VISID):	Uniquely identifies the visualisation area
TripID (FahrtID):	(sub-element) Identifies a trip for visualisation
LineID (LinienID):	Line code of the trip
LineText (LinienText):	Name of the line (for the passenger)
DirectionID (RichtungsID):	Direction code of the trip
DirectionText (RichtungsText):	Name of the direction (for the passenger)
TripStatus (FahrtStatus):	Specifies whether real-time information can be supplied for the vehicle (“real”) or not (“schedule”). If only scheduled information is available, positions and delays should be considered as old.
ActStop (AktHst):	(optional) Short name of the current or last stop
StopSeqCount (HstSeqZaehler):	Must be specified in conjunction with ActStop. Specifies the passage sequence in the case of circular trips.
AtStop (AufHst):	(optional) Flag that indicates whether a vehicle is currently at the ActStop (“true”).
NextStop (NachHst):	(optional) Short name of the stop, which the trip arrives at next
Delay (Verspaetung):	(optional) Current delay in seconds, early times are shown as nega
StarttStop (StartHst):	(optional) Short name of the start stop
EndStop (EndHst):	(optional) Short name of the end stop
Distance (Distanz):	(optional) Distance travelled as a percentage of the section between ActStop and NextStop.
Longitude (Longitude):	(optional) Geographic longitude WGS-84 (milliseconds).
Latitude (Latitude):	(optional) Geographic latitude in WGS-84 (milliseconds).
QueueIndicator (Stauindikator):	(optional) Specifies whether the vehicle is in a traffic jam (true) or not (false).
TripInfo (FahrtInfo):	(optional, sub-element) More information on the trip

The following example shows a response to the above subscription. One vehicle is reported.

```
<DataSupplyAnswer>
      <Acknowledge
            TimeStamp="2001-08-08T05:05:10"
            Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <VISMessage SubscriptionID="25">
            <VISDeviation      TimeStamp="2001-08-08T05:05:05"
                        ValidUntilTimeStamp="2001-08-08T05:05:05">
                  <VISID>12345</VISID>
                  <TripID>
                        <TripName>64356</TripName>
                        <DayType>2001-08-08</DayType>
                  <LineID>10</LineID>
                  <LineText>X10</LineID>
                  <DirectionID>Zoo</DirectionID>
                  <DirectionText>Zoological Garden</DirectionID>
                  <TripStatus>Real</TripStatus>
                  <ActStop>PARISSTR</ActStop>
                  <StopSeqCount>1</StopSeqCount>
                  <NextStop>NBGRSTR</NextStop>
                  <Distance>50</Distance>
                  <Longitude>46800123</Longitude>
                  <Latitude>93355177</Latitude>
                  <QueueIndicator>false</QueueIndicator>
            </VISDeviation>
      </VISMessage>
</DataSupplyAnswer>
```

6.4.3.3.2 Deleting Visualisation Data (VISTripDelete)

If the trip leaves the visualisation area (normal situation is the completion of the trip), the data producer must send a deletion message, which prompts the displaying system to remove the trip from the corresponding representations.

For this it supplies a *VISMessage* within a *DataSupplyAnswer* with the *VISTripDelete* element. There are no more transmissions of *VISDeviation*.

Definition of *VISTripDelete (VISFahrtLoeschen)*:

TimeStamp (Zst):	(attribute) The time stamp registers the time of data recording.
VISID (VISID):	Uniquely identifies the visualisation area
TripID (FahrtID):	(sub-element) Identifies the trip to be deleted
LineID (LinienID):	Line code of the trip
DirectionID (RichtungsID):	Direction code of the trip
LineText (LinienText):	Name of the line (for the passenger)
DirectionText (RichtungsText):	Name of the direction (for the passenger)
Reason (Ursache):	(optional) Reason for unexpected departure from the visualisation area (e.g. premature cancellation of the trip). Omitted under normal circumstances (end of trip).

A message that deletes the above trip (65356) could look as follows:

```
<DataSupplyAnswer>
      <Acknowledge TimeStamp="2001-08-08T05:25:10" Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <VISMessage SubscriptionID="25">
      <VISTripDelete TimeStamp="2001-08-08T15:25:00">
            <VISID>12345</VISID>
            <TripID>
                  <TripName>64356</TripName>
                        <DayType>2001-08-08</DayType>
            </TripID>
            <LineID>10</LineID>
            <DirectionID>Zoo</DirectionID>
            <LineText>X10</LineText>
            <DirectionText>Zoological Garden</DirectionText>
            <Reason>Engine failure</Reason>
      </VISTripDelete>
      </VISMessage>
</DataSupplyAnswer>
```

6.5 General Message Service (GMS)

6.5.1 Introduction

The message service is used to transmit general text information between the participant control centres. As with the other services, it is based on the client-server model. The server is the system that wishes to send data. The client is the recipient of these text messages. The message service differentiates between different information channels. These can be assigned to operationally different message types (errors, messages, warnings, traffic information, operational messages, etc.) Each channel can be polled separately.

Messages cannot only be sent, but also revoked. This makes sense where a message loses its validity before the planned time. If, for example, a message has been sent to set up a special purpose vehicle with a planned duration, the message can be revoked with a deletion message if this vehicle finishes early.

The message service only involves an exchange of process data. There is no need for an exchange of reference data.

6.5.2 Message Formats

The message service can basically transmit three types of data:

→ Simple texts

→ Texts with structural characteristics

→ Freely defined XML content

Whilst XML contents have an inherent structure, formatting can be applied to simple texts, which permits subsequent structuring. CSV (comma separated value) files represent an example of such a format. It is a table format in which the column values are separated by commas and the rows by line-feed characters.

In order to be able to use and detect different formats at run time, a so-called *FormatID* is used to highlight the format in every message. For XML content the URL of the given schema should be used as the *FormatID*. If no FormatID is specified, then by definition the message is a simple text without formatting (can be displayed directly).

Every implementation of a message service must at least support this format.

6.5.3 Operational Data Supply and Management

In order to be able to agree the different classes among the operations, it is necessary to define so-called *ChannelIDs*. These define a common understanding of the message classes. The *ChannelIDs* are managed within the data management of the participant systems. The agreement is bilateral.

The following example shows the definition of two message channels for the communication of messages to operation AVMS B:

ChannelID	Foreign operator code	Message class (internal)
1	AVMS B	Errors
2	AVMS B	Other messages

Table 17: Definition of message channels

The *FormatIDs* must also be agreed on by the various operators, if structured data is to be exchanged between the control centres.

6.5.4 Process Data Service

As in the other services, communication begins with the setting up of a subscription. In the message service, this is achieved by the transmission of a *GMSSubscription* message contained within a *SubscriptionRequest* (6.5.4.1).

In *GMSSubscription* it is possible to specify one or more channels, whose data is to be polled.

If there is a message waiting to be transmitted, it is signalled by the server with a *DataReadyRequest*. After acceptance and confirmation, the messages are then retrieved by the client with a *DataSupplyRequest*. The server supplies its information within *GMSMessage* ***(Fehler! Verweisquelle konnte nicht gefunden werden.)***. The message contains the actual information in *GMSNotification (6.5.4.2)* elements. These have their own ID and validity (*ValidUntilTimeStamp*). Via the ID of a *GMSNotification* it is possible to achieve specific deletion (*GMSNotificationDelete* message, 6.5.4.2.2).

6.5.4.1 Polling Messages (*GMSSubscription*)

Message polling is initiated by the client. It sets up a *SubscriptionRequest* with one or more *GMSSubscription* sub-elements

Definition of *GMSSubscription (AboAND)*:

ValidUntilTimeStamp (VerfallZst):	(attribute) Specifies the time to which the subscription is valid
SubscriptionID (AboID):	(attribute) The SubscriptionID references the message subscription as created by the request. It is assigned by the requesting system.
ChannelID (KanalID):	(multiple) Message channel whose messages are to be polled

The following example illustrates a request that polls two data channels between 5:00 a.m. and 10:00 p.m.:

```
<SubscriptionRequest Sender="AVMS A" TimeStamp="2001-08-08T05:00:00">
      <GMSSubscription SubscriptionID="25" ValidUntilTimeStamp="2001-08-
08T22:00:00">
      <ChannelID>Disruptions</ChannelID>
      <ChannelID>Passenger accidents</ChannelID>
      </GMSSubscription>
</SubscriptionRequest>
```

The server confirms the successful creation of the subscription with an *Acknowledge* element in a *SubscriptionReply*.

6.5.4.2 Messages in the Message Service (*GMSMessage*)

All messages sent from the server to the client are encapsulated within a *GMSMessage* element. The *GMSMessage* therefore defines the frame of the subscription.

GMSMessage can contain the following sub-elements:

→ Send message (*GMSNotification*)

→ Cancellation of a text message (*GMSNotificationDelete*)

Definition of *GMSMessage (ANDNAchricht)*:

SubscriptionID (AboID):	(attribute) The SubscriptionID references the message subscription as created by the request. It is assigned by the requesting system.
GMSNotification (ANDMeldung):	(sub-element, optional, multiple) Contains the useful message data
GMSNotificationDelete (ANDMedlungLoeschen):	(sub-element, optional, multiple) Instructs the receiver to remove the message, or to mark it as invalid / expired.

6.5.4.2.1 Transferring Messages (*GMSNotification*)

If the server has messages waiting to be sent, it sends a *DataReadyRequest* to the client. This confirms reception with an *Acknowledge* in a *DataReadyAnswer*.

The client now requests the data with a *DataSupplyRequest*. The server responds with a *DataSupplyAnswer*, which contains one or more *GMSMessage* elements.

The message information is contained within *GMSMessage* in *GMSNotification* elements. These contain the actual useful data, have their own unique ID (*NotificationID*) and a life span (*ValidUntilTimeStamp*). They can be updated (overwritten) or declared prematurely invalid.

The *NotificationID* is unique for the entire GMS service (across channel borders).

Definition of *GMSNotification (ANDMeldung)*:

TimeStamp (Zst):	(attribute) The time stamp registers the time the message was created
ValidUntilTimeStamp (VerfallZst):	(attribute) Specifies the planned time to which the message is operationally valid (e.g. end of an action).
ChannelID (KanalID):	Uniquely identifies the message channel
NotificationID (MeldungsID):	Identifies the message within the service. It is used to over-write or delete messages
FormatID:	(optional) Uniquely identifies the text format
MessageXML:	(optional) contains messages in xml format

The following example illustrates the transmission of two messages for the above subscription. Simple text without formatting is used:

```
<DataSupplyAnswer>
      <Acknowledge
            TimeStamp="2001-08-08T10:00:00"
            Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <GMSMessage SubscriptionID="25">
            <GMSNotification  TimeStamp="2001-08-08T9:59:30"
                                    ValidUntilTimeStamp="2001-08-
08T11:00:00">
            <NotificationID>4711</NotificationID>
            <ChannelID>Disruptions</ChannelID>
            Two-lane block on High Bridge due to roadwork.
            </GMSNotification>
            <GMSNotification   TimeStamp="2001-08-08T9:58:00"
                         ValidUntilTimeStamp="2001-08-11T12:00:00">
            <ChannelID>Disruptions</ChannelID>
            <NotificationID>4712</NotificationID>
            Three-day obstruction of the inner ring road.
            </GMSNotification>
      </GMSMessage>
</DataSupplyAnswer>
```

The following example illustrates the use of formatted text (CSV):

```
<DataSupplyAnswer>
      <Acknowledge
            TimeStamp="2001-08-08T10:00:00"
            Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <GMSMessage SubscriptionID="25">
            <GMSNotification  TimeStamp="2001-08-08T9:59:30"
                                    ValidUntilTimeStamp="2001-08-
08T11:00:00">
                  <ChannelID>Disruptions</ChannelID>
                  <NotificationID>4711</NotificationID>
                  <FormatID>CSV</FormatID>
                  Two-lane block,High Bridge,roadwork,
            </GMSNotification>
            <GMSNotification  TimeStamp="2001-08-08T9:58:00"
                        ValidUntilTimeStamp="2001-08-11T12:00:00">
            <NotificationID>4712</NotificationID>
            <ChannelID>Disruptions</ChannelID>
                  <FormatID>CSV</FormatID>
            Obstruction, inner ring road, ,3 days.
            </GMSNotification>
      </GMSMessage>
</DataSupplyAnswer>
```

And the same message again with an XML format:

```
<DataSupplyAnswer>
      <Acknowledge
            TimeStamp="2001-08-08T10:00:00"
            Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <GMSMessage SubscriptionID="25">
            <GMSNotification  TimeStamp="2001-08-08T9:59:30"
                                    ValidUntilTimeStamp="2001-08-
08T11:00:00">
                  <ChannelID>Disruptions</ChannelID>
                  <NotificationID>4711</NotificationID>
                  <FormatID>www.myschema.com/ANDSchema</FormatID>
                  <MessageXML>
                  <Incident> Two-lane block </Incident>
                  <Location>High Bridge</Location>
                  <Reason>Roadwork</Reason>
                  </MessageXML>
            </GMSNotification>
            <GMSNotification  TimeStamp="2001-08-08T9:58:00"
                        ValidUntilTimeStamp="2001-08-11T12:00:00">
                  <NotificationID>4712</NotificationID>
                  <ChannelID>Disruptions</ChannelID>
                  <FormatID>www.myschema.com/ANDSchema</FormatID>
                  <MessageXML>
                  <Incident>Obstruction</Incident>
                  <Location>Inner ring road</Location>
                  <Duration>3 days</Duration>
                  </MessageXML>
            </GMSNotification>
      </GMSMessage>
</DataSupplyAnswer>
```

If a new *GMSNotification* is sent, it overwrites the content of the old message. This allows the validity of the messages to be subsequently changed.

6.5.4.2.2 Deleting Messages (GMSNotificationDelete)

If a message loses its validity before expiration of ValidUntilTimeStamp, the server can inform the client with the transmission of a *GMSNotificationDelete* message. It is not possible to update (overwrite) a previously deleted message.

The referencing of the message to be deleted is achieved via the *NotificationID*.

Definition of *GMSNotificationDelete (ANDMeldungLoeschen)*:

TimeStamp (Zst):	(attribute) The time stamp registers the time at which the deletion message was recorded.
NotificationID (MeldungsID):	Identifies the message to be deleted within the service

The following example revokes message with ID 4711:

```
<DataSupplyAnswer>
      <Acknowledge
            TimeStamp="2001-08-08T10:00:00"
            Result="ok"
            ErrorNumber="0">
      </Acknowledge>
      <PendingData>false</PendingData>
      <GMSMessage SubscriptionID="25">
            <GMSNotificationDelete  TimeStamp="2001-08-08T9:59:30">
                  <NotificationID>4711</NotificationID>
            </GMSNotificationDelete>
      </GMSMessage>
</DataSupplyAnswer>
```

7 Glossary

The glossary serves to ensure a common use of terms within this project, whereby all specifications and definitions, which are necessary for processing, are based on the terms contained herein. Any necessary continuations are carried out in the course of ongoing project management.

Term	Description
AVMS	Automatic Vehicle Management System
Changeover time	Time needed to change between vehicles at a connection point
Connection planning	Definition of the connections to be monitored (on the basis of the day's schedule)
Connection point (stop)	Stop or station at which several modes of transport vehicles stop for the purpose of connections
Control centre	Set-up to control and monitor the traffic and operational procedures of a transport authority
CP	Connection protection: Service for the operational exchange of data for connection protection
Delay	Positive deviation from planned schedule
Dispatch	Operative management for controlling the traffic and operation
Display owner	AVMS that controls the displays
Display text	Displayed information about line, trip destination, departure time as well as special texts and service information
Display user	AVMS that displays its trips on the displays of another AVMS
DPI	Dynamic passenger information: Service for the operational exchange of data for the purpose of passenger information
Early time	Negative deviation from planned schedule
Feeder (trip)	Trip that brings passengers to a connection point
Fetcher (trip)	Trip that takes on passengers from a feeder vehicle at a connection point
GMS	General message service: Service to exchange operational messages between two control centres
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
Line trip	Trip of a line
Meta data	The definitions and fundamentals agreed by two transport authorities as the basis for data exchange
Operating day (day type)	Time period for the validity of schedules within an AVMS (may be different for different operations)
Process data exchange	Exchange of real-time information between two AVMS
Published schedule	Schedule that is posted at stations and stops; it illustrates the services on offer
Real-time (real) schedule	Schedule generated from the planned schedule that has been complemented with real-time information
REF, reference data exchange	Exchange of the planned schedules between two AVMS

Term	**Description**
Route	Path on a line (from A to B); a line can comprise several routes (e.g. from A to B via C)
Run vehicle	Section of a train that is separated from the remainder of the trainset during the course of the trip and then continues on its own or within a new trainset.
Schedule prediction	Preview of the real-time schedule for a future time period
Special trip	Trip that encompasses one or more lines, which does not necessarily serve all stops
Stop name	Unique identification characteristic of a stop (e.g. stop number)
Subscription procedure	Established means of communication in the interface for the purpose of data exchange
Trainset	Assembly of coupled single trips (see run vehicle).
Trip	Translation of the german term “Fahrt”. In the wording of the European Data Model “Transmodel” and SIRI as well the right word would be “(vehicle) jour-ney”.
Trip announcement	Generation and display of trip information on the basis of the schedule data
Trip name	Unique identification characteristic of a trip (e.g. trip number, trip ID)
Validity time frame of schedules	Defining time period for which a schedule is valid, e.g. a schedule period. Dif-ferent transport operators do not usually have matching validity periods for schedules.
Via texts	Text that informs the passengers about the course of a trip (important interme-diate stops)
VIS	Visualisation: Service for the exchange of process data for the purpose of visu-alising foreign vehicles in a control centre
XML	Extended Markup Language

8 References

HTTP ?2.6.2001

http://www.w3.org/Protocols/
Web page of the W3 consortium on HTTP. Overview page for W3C documents on HTTP and references to other sources.

HTTP/1.1-Specification June 1999

http://www.ietf.org/rfc/rfc2616.html
RFC2616 specification of the Hypertext Transport Protocol (HTTP) of the IETF.

XML 27.10.2001

http://www.w3.org/XML/
Web page of the W3 consortium on XML. Overview page for W3C documents on XML and references to other sources.

XML in 10 points **27.10.2001**

http://www.w3.org/XML/1999/XML-in-10-points
Summary of the meaning of XML.

XML Schema Part 0: Primer (Recommendation) **25.05.2002**

http://www.w3.org/TR/xmlschema-0/
Non-normative, comprehensible introduction into the capabilities of the XML schema, an XML-based mark-up language, through which along with DTDs it is possible to define valid XML data structures.
http://www.edition-w3c.de/TR/2001/REC-xmlschema-0-20010502/

XML Schema Part 1: Structures **25.05.2002**

http://www.w3.org/TR/xmlschema-1/
Part 1 of the normative description of the XML schema. This section describes the layout and the elements of the XML schema.
http://www.edition-w3c.de/TR/2001/REC-xmlschema-1-20010502/

XML Schema Part 2: Data Types **25.05.2002**

http://www.w3.org/TR/xmlschema-2/
Part 2 of the normative description of the XML schema. This section describes the elementary data types of the XML schema.
http://www.edition-w3c.de/TR/2001/REC-xmlschema-2-20010502/

9 English Aliases

The following table lists the original German terms alongside the accepted English aliases for VDV453, which are used in international implementations.

9.1 Services

VDV 453		**SIRI Equivalents**		**VDV 453 English Aliases**	
Name	abbr	Name	abbr		
Anschluss-Sicherung Referenzdatendienst	REF-ANS	Connection Timetable Service	CT	reference data connection protection service	REF-CP
Anschluss-Sicherung Prozessdatendienst	ANS	Connection Monitoring Service	CM	process data connection protection service	CP
Dynamische Fahrgastinformation Referenzdatendienst	REF-DFI	Stop Timetable Service	ST	Reference Data Service	REF-DPI
Dynamische Fahrgastinformation Prozessdatendienst	DFI	Stop Monitoring Service	SM	Process Data Service	DPI
Visualisierung	VIS	Vehicle Monitoring Service	VM	Visualisation of Foreign Vehicles	VIS
Allgemeiner Nachrichtendienst	AND	General Message Service	GM	General Message Service	GMS

9.2 Root-Elements and complex Subelements

VDV 453	SIRI	VDV 453 English Aliases
AbbringerFahrtLoeschen	DistributorDepartureCancellation	FetcherTripDelete
AbbringerInfo	DistributorInfoGroup	FetcherInfo
Abbringernachricht	Connection MonitoringDistributorDelivery	FetcherMessage
AboAND	GeneralMessageSubscriptionRequest withGeneralMessageRequest	GMSSubscription
AboAnfrage	SubscriptionRequest	SubscriptionRequest
AboAntwort	SubscriptionResponse	SubscriptionReply
AboASB	ConnectionMonitoringSubscriptionRequest	CPISubscription
AboASBRef	ConnectionTimetableSubscriptionRequest with ConnectionTimetableRequest	CPIRefSubscription
AboAZB	StopMonitoringSubscriptionRequest, StopMonitoringRequest	DISSubscription
AboAZBRef	StopTimetableSubscriptionRequest with StopTimetableRequest	DISRefSubscription
AboLoeschen	TerminateSubscriptionRequest	DeleteSubscription
AboVIS	VehicleMonitoringSubscriptionRequest withVehicleMonitoringRequest	VISSubscription
ANDMeldung	InfoMessage	GMSNotification
ANDMeldungLoeschen	InfoMessageCancellation	GMSNotificationDelete
ANDNachricht	GeneralMessageDelivery	GMSMessage
ASBFahrplan	ConnectionTimetableDelivery with TimetabledFeederArrival	CPISchedule
ASBFahrplanlage	MonitoredFeederArrival	CPIDeviation
ASBFahrtLoeschen	MonitoredFeederArrivalCancellation	CPITripDelete
AZBFahrplan	StopTimetableDelivery with TimetabledStopVisit	DISSchedule
AZBFahrplanlage	MonitoredStopVisit	DISDeviation
AZBFahrtLoeschen	MonitoredStopVisitCancellation	DISTripDelete
AZBLinienspezialtext	StopLineNotice	DISLineSpecialText
AZBLinienspezialtextLoeschen	StopLineNoticeCancellation	DISLineSpecialTextDelete
AZBNachricht	StopTimetableDelivery (ST Service) StopMonitoringDelivery (SM Service)	DISMessage
Bestaetigung	ResponseStatus	Acknowledge
DatenAbrufenAnfrage	DataSupplyRequest	DataSupplyRequest
DatenAbrufenAntwort	ServiceDelivery	DataSupplyAnswer
DatenBereitAnfrage	DataReadyNotification	DataReadyRequest
DatenBereitAntwort	DataReadyAcknowledgement	DataReadyAnswer
Direktruf	./. (to be added in VehicleJourneyInfo)	Direct Call
FahrtFilter	ConnectingJourneyFilter	TripFilter

VDV 453	**SIRI**	**VDV 453 English Aliases**
FahrtID	FramedVehicleJourneyRef	TripID
FahrtInfo	VehicleJourneyInfo	TripInfo
HaltepositionsAenderung	StoppingPositionChangedDeparture	StopPositionChange
Status	Status	Status
StatusAnfrage	CheckStatusRequest	StatusRequest
StatusAntwort	CheckStatusResponse	StatusReply
Traktion	TrainBlockPart	Trainset
VISFahrplanlage	VehicleActivity	VISDeviation
VISFahrtLoeschen	VehicleActivityCancellation	VISTripDelete
VISNachricht	VehicleMonitoringDelivery	VISMessage
WartetBis	WaitProlongedDeparture	WaitUntil
ZeitFilter	ConnectingTimeFilter	TimeFilter
Zubringernachricht	ConncetionTimetableDelivery (CT Service) ConnectionMonitoringFeederDelivery (CM Service)	FeederMessage

9.3 Other Elements

It is recommended to view the SIRI documentation to identify possible deviations of the definitions: www.siri.org.uk

VDV 453	SIRI	VDV 453 English Aliases
AbfahrtszeitASBPlan	AimedDepartureTime (CM Service)	ScheduledCPIDepartureTime
AbfahrtszeitASBPrognose	ExpectedDepartureTime (CM Service)	ExpectedCPIDepartureTime
AbfahrtszeitAZBPlan	AimedDepartureTime (SM Service)	ScheduledDISDepartureTime
AbfahrtszeitAZBPrognose	ExpectedDepartureTime (SM Service)	ExpectedDISDepartureTime
AbfahrtszeitAZB-Disposition	./. (element 'ConditionalExpected-DepartureTime' to be added in MonitoredStopVisit)	AimedDISDepartureTime
AbfahrtszeitStartHst	OriginAimedDepartureTime	DepartureTimeStartStop
AbmeldeID	CleardownRef	DepartureNoticeID
AboID	SubscriptionRef with Subscription-Identifier	SubscriptionID
AboLoeschenAlle	All (inTerminateSubscriptionRequest)	DeleteSubscriptionsAll
AktHst	StopPointNameRef (in Monitored-Call)	ActStop
AnkunftszeitASBPlan	AimedArrivalTime (CM Service)	ScheduledCPIArrivalTime
AnkunftszeitASBPrognose	ExpectedArrivalTime (CM Service)	ExpectedCPIArrivalTime
AnkunftszeitAZBPlan	AimedArrivalTime (SM Service)	ScheduledDISArrivalTime
AnkunftszeitAZBPrognose	ExpectedArrivalTime (SM Service)	ExpectedDISArrivalTime
AnkunftszeitZielHst	Destination AimedArrivalTime	ArrivalTimeDestinationStop
AnzahlFahrten	NumOfBlockParts	NumOfTrips
ASBID	ConnectionLinkRef	CPIID
AufASB	VehicleAtStop (CM Service)	AtCPIPoint
AufAZB	VehicleAtStop (SM Service)	AtDISPoint
Aufgabe	Cancellation	Cancelled
AufHst	VehicleAtStop	AtStop
AZBID	MonitoringRef	DISID
Betreiber	OperatorRef	Operator
BetrieblicheFahrzeug-nummer	VehicleRef (in EstimatedVehicle-Journey /OperationalInfoGroup)	VehicleNumber
Betriebstag	DataFrameRef	DayType
DatenBereit	Ref. 9.2: Status	DataAvailable
Datengueltigbis	ValidUntil	DataValidUntil
DatensatzAlle	AllData (in DataSupplyRequest)	AllData
Distanz	Percentage	Distance
EndHst	DestinationShortName	EndStop
Ergebnis	Status (in SubscriptionResponse)	Result
FahrtBezeichner	DatedVehicleJourneyRef	TripName
Fahrtspezialtext	StopVisitNote	TripSpecialText
FahrtStatus	Monitored	TripStatus
FahrzeugID	VehicleRef	VehicleID
Fehlernummer	ErrorCondition	ErrorNumber
Fehlertext	Description (in ResponseStatus)	Errortext
FormatID	FormatRef	FormatID

VDV 453	SIRI	VDV 453 English Aliases
FruehesteAbfahrtszeit	StartTime (DepartureWindow in StopTime-tableRequest)	EarliestDepartureTime
FruehesteAnkunftszeit	StartTime (ArrivalWindow in ConnectionTi-metableRequest) EarliestArrivalTime (ConnectingTimeFilter in ConnectionTimetableMonitoring))	EarliestArrivalTime
HaltepositionsText	ChangeNote, NewLocation	StopPositionText
HaltID	StopPointRef	StopID
HstSeqZaehler	VisitNumber	StopSeqCount
Hysterese	ChangeBeforeUpdate	Hysteresis
IP-Adresse	./. (to be added in VehicleJourneyInfo)	IP-Adress
KanalID	InfoChannelRef	ChannelID
KuerzMoeglicherZyklus	ShortestPossibleCycle	ShortestPossibleCycleTime
KursNr	CourseOfJourneyRef	RunNumber
Latitude	Latitude (in: VehicleLocation)	Latitude
LinienID	LineRef	LineID
LinienNr	ExternalLineRef	LineNumber
Linienspezialtext	LineNote	LineSpecialText
LinienText	PublishedLineName	LineText
Longitude	Longitude (in: VehicleLocation)	Longitude
MaxAnzahlFahrten	MaximumNumberOfCalls	MaxNumOfTrips
MaxTextLaenge	MaximumTextLength	MaxTextLength
MeldungsID	InfoMessageIdentifier	NotificationID
NachHst	StopPointRef (in OnwardCall)	NextStop
Position	PositionOfTrainBlockPart	Position
Prioritaet	./. (new element to be added in	Priority
ProduktID	ProductCategoryRef	ProductID
Prognose	Estimation	Prediction
RichtungsID	DirectionRef	DirectionID
RichtungsText	DirectionName	DirectionText
Sender	SubscriberRef	Sender
ServiceMerkmal	ServiceFeatureRef	ServiceAttribute
SpaetesteAbfahrtszeit	EndTime (DepartureWindow in StopTime-tableRequest)	LatestDepartureTime
SpaetesteAbbringerInfo	./. (new element to be added in MonitoredFeederArrival)	LatestFetcherInfoTime
SpaetesteAnkunftszeit	EndTime (ArrivalWindow in ConnectionTi-metableRequest) LatestArrivalTime (ConnectingTimeFilter in ConnectionTimetableMonitoring)	LatestArrivalTime
Sprachausgabe	./. (New element 'TextToSpeach' to be ad-ded in StopLineNotice)	SpeechOutput
StartDienstZst	ServiceStartedTime	StartServiceTimeStamp
StartHst	OriginRef (als Identifier) OriginShortName (als Kurzbezeichner)	DepartureStop
StartHstLang	OriginName	DepartureStopLong
Stauindikator	InCongestion	QueueIndicator
Telefonnummer	./. (to be added in VehicleJourneyInfo)	Telephonenumber
TraktionsID	TrainPartRef	TrainsetID
UmlaufNr	BlockRef	BlockNumber
UmsteigeWillige	NumberOfTransferPassengers	TransferPassengers
Ursache	Reason	Reason
VerfallZst	InitialTerminationTime, ValidUntilTime	ValidUntilTimeStamp
Verlaesslichkeit		Reliability

VDV 453	**SIRI**	**VDV 453 English Aliases**
Verspaetung	Delay	Delay
ViaHst	Vianame	ViaStop
VISID	VehicleMonitoringRef	VISID
Vorschauzeit	PreviewIntervall	PreviewTime
WeitereDaten	MoreData	PendingData
ZielHst	DestinationShortName	DestinationStop
ZielHstLang	DestinationName	DestinationStopLong
Zst	RecordedAtTime	TimeStamp
ZubringerHstLang	StopPointName	FeederStopLong
Zyklus	UpdateIntervall	Cycle