

GIVA Platform requirements

Specification

Confidentiality:

Public

Version control

Version	Date	Remarks
2.3	29-10-2019	Final version for applications and devices

Referenced documents

Reference	Document
1	VDV 301-1 Internet Protocol based Integrated On-Board Information 01/2014 Part 1 – System architecture
2	VDV 301-2 Internet Protocol based Integrated On-Board Information 01/2014 Part 2 – Interface Specification
3	EN13149-7 Public transport - Road vehicle scheduling and control systems - Part 7: System and Network Architecture July 2015



Table of Contents

1	Introduction	4
2	GIVA applications.....	4
2.1	GIVA platform	4
2.2	GIVA application requirements	5
2.2.1	Service discovery	6
2.2.2	Logging.....	7
2.2.3	Service/application-specific web interface	7
2.2.4	Starting and stopping	8
2.2.5	Service interfaces.....	8
2.2.6	Service directory.....	10
2.2.7	Data.....	10
2.2.8	Information exchange	10
2.2.9	Platform and development	12
3	GIVA peripheral devices.....	14
3.1	Physical requirements	14
3.2	Functional requirements.....	14
3.2.1	Service discovery	15
3.2.2	Monitoring.....	15
3.2.3	Device-specific web interface.....	16
3.2.4	Starting and stopping	16
3.2.5	Service interface consumers.....	16
3.2.6	Information exchange	17
3.2.7	Time synchronisation	17
4	Appendix – best practices	18



1 Introduction

This document describes the Generic ICT Vehicle Architecture (GIVA) used in the GVB vehicles, specifically the requirements for applications and (peripheral) devices to comply to the GIVA platform which manages the starting/stopping of the (complete) system, configuration of the system, monitoring, logging, etc.

- Chapter 2 describes the requirements for GIVA applications
- Chapter 3 describes the requirements for (peripheral) devices, connecting to the GIVA platform.

2 GIVA applications

GIVA applications are applications running on the generic GIVA computer and using the GIVA (software) platform.

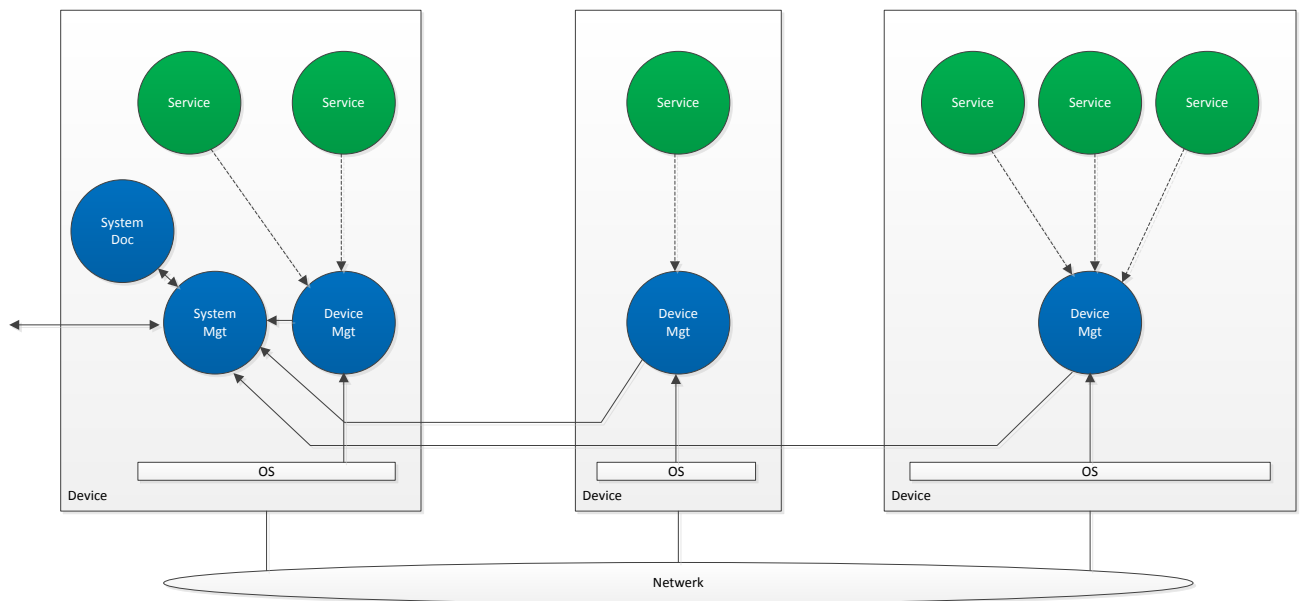
2.1 GIVA platform

All functions/devices on the GIVA infrastructure must use the GIVA platform for configuration, management and monitoring.

This concerns:

- Service discovery
- System management
- System configuration and logging
- Device management
- Monitoring
- Logging
- Configuration management
- Time synchronisation

GIVA is based on norm EN13149 and uses the implementation conform ITxPT and VDV301 (IBIS-IP). Important functions within VDV301 are the SystemManagement and DeviceManagement services. These services manage the starting, stopping and monitoring of the complete vehicle system. The figure below gives an overview of the services and their relations.



2.2 GIVA application requirements

Applications must comply to a subset of the functionality of the GIVA platform, regarding:

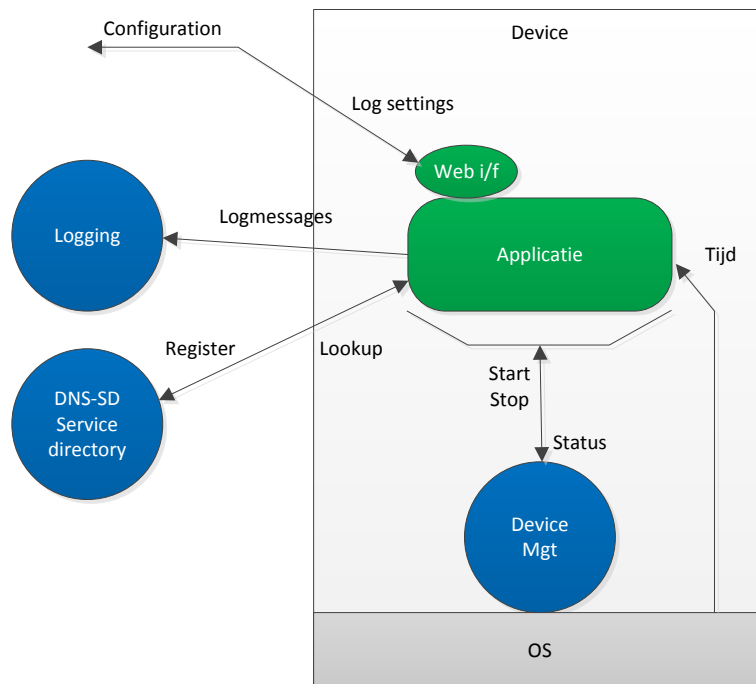
- **Service discovery**
For registering the supplied services and discovering the services, needed to provide the required functionality.
- **Logging**
To register status information from an application on a central location in the vehicle.
- **Device management**
Applications have no *functional* relation with Device Management, this function manages the starting/stopping and monitoring of the application.
Each application must provide a Device Management function.

Additionally, each application must provide an application/service specific web interface for:

- Verifying the functionality of the application.
- Providing configuration information such as content/data versions, etc.

Finally, application must comply to a number of generic requirements regarding implementation, platform, performance, etc.

The figure below gives an overview of the relations.



2.2.1 Service discovery

Essential function within a flexible, service oriented environment, is the setup of communication between services (on random physical devices) and peripheral devices.

In order to implement a 'zero-configuration' mechanism without static configuration, GIVA uses multicast DNS and DNS-SD for publication and discovery of services.

[E-001] Each application must (a) publish its service interfaces and (b) discover service interfaces, using DNS-SD on mDNS conform document EN13149 Part 7 [3].

[E-002] Each application must be resistant to start-up behaviour of the service directory, random starting order of service and restarting of services.

Exceptions and additional information.

In order to limit the DNS-SD dependency, each service consumer must (persistently) store the connection information of required services. Upon starting a consumer, it should first discover services in the service directory; if the service directory is not available, the consumer should use the stored information as a fall-back.

[E-003] Each application must store the connection information of required services when these connections are set up.

[E-004] Each application must, upon starting, if the DNS-SD service directory is not available after a timeout, use the stored connection information to connect to the required services.



2.2.2 Logging

Each application must register information about events in the application at one central location in the vehicle. This storage functionality is provided by the SystemDocumentation service conform VDV301 documents [1] en [2].

[E-010] Each application must send logging information to the SystemDocumentation service using StoreLogMessages operations conform VDV301 documents [1] en [2].

[E-020] Each application must send logging information to the SystemDocumentation service formatted in agreement with GVB.

[E-030] Each application must be able to buffer logging information for at least 1 minute to handle start-up/restart behaviour of the SystemDocumentation service.

2.2.3 Service/application-specific web interface

This web interface is intended for viewing and setting application-specific configuration and status information.

[E-040] Each application must offer a web interface for viewing and setting application-specific configuration and status information.

[E-041] Each application must publish the connection information of the web interface in the Service directory.

[E-042] The web interface must be able to report the current list of subscriptions for all service interfaces it provides.

[E-043] The web interface must be able to report the versions of the application content (media, timetable, etc.)

2.2.4 Starting and stopping

[E-050] Upon starting/stopping, the application must send concerning log messages to the System Documentation service conform [4] to reflect its status.

[E-051] Upon starting, an application must discover its required services in the Service Directory. Depending on the availability of these services, the application must determine which functionality it can provide and if service interfaces can be published. Determining this 'degraded mode' functionality is part of the application design and must be agreed with GVB.

[E-052] Upon service discovery, subscription to services, etc. each application must send log messages to the System Documentation service conform [4] to reflect the current state of each connection.

2.2.5 Service interfaces

[E-061] Service interfaces must validate all incoming messages to an XML Schema Definition (XSD) for the concerning interface.

[E-062] Service interfaces must handle corrupt data and/or data which does not comply to the XML Schema Definition (XSD), without any impact on its normal operation.

[E-063] Service interfaces must, upon receiving corrupt data and/or data which does not comply to the XML Schema Definition (XSD), send a log message to the System Documentation service conform [4]. As soon as correct data is received again, a log message must be sent to indicate the end of the error situation.

[E-064] Service interfaces must respond on incoming messages within 1 second for 95% of the cases.

[E-065] Service interfaces supporting subscriptions must function correctly with 10 subscribers and information changing every second.

[E-066] Service interfaces must function correctly with 10 consumers requesting information at a total of 20 times per second.

Service interfaces - Subscription

[E-070] Service interfaces supporting subscriptions must be able to handle subscription requests containing the return information of the subscriber.

[E-071] Service interfaces supporting subscriptions must answer subscription requests synchronously with a subscription response.

[E-072] Service interfaces supporting subscriptions must, after sending a positive subscription response, automatically send an information frame to the subscriber endpoint.



- [E-073] Service interfaces supporting subscriptions must send a message with updated information to each subscriber when the subscribed information changes.
- [E-074] Service interfaces supporting subscriptions must be able to handle unsubscription requests by removing the concerning subscription.
- [E-075] Service interfaces supporting subscriptions must answer unsubscription requests synchronously with an unsubscription response.
- [E-076] Service interfaces supporting subscriptions must, upon stopping the concerning application, send a subscription response with status false (unsubscription response) to each subscriber to end its subscription.
- [E-077] Service interfaces supporting subscriptions must support only one subscription for an individual subscriber. Additional subscription requests from the same subscriber must be handled correctly but ignored in its internal administration.
- [E-078] Messages from one service interface to multiple subscribers must be sent with minimal time difference. This implies that these messages must all be handled independently..
- [E-079] Service interfaces supporting subscriptions must support at least 20 simultaneous subscriptions.
- [E-080] As a keep-alive mechanism, each Service Interface sends a message to each Subscriber every minute (60 seconds) conform a normal Get.
- [E-081] If a Subscriber does not receive a message within one minute (60 seconds), it should handle this as follows:
- Unsubscribe at the concerning Service interface
 - Check in DNS-SD if the endpoint is still available or has changed.
 - Subscribe.
 - Repeat the previous 2 steps every 2 seconds.
 - Send a log message to the System Documentation on each state change in previous steps.
- [E-082] It is not allowed to periodically subscribe to a Service interface if a subscription is already active.
- [E-083] If a Service interface sends a message to a subscriber and this message is not acknowledged, this should be handled as follows:
- Retry once after 2 seconds.
 - Check in DNS-SD if the endpoint is still available or has changed.
 - Retry again for 3 times every 2 seconds.
 - Send a log message to the System Documentation on each state change in previous steps.



Service interfaces – Get/Post

[E-090] If a Service interface sends a response to a Get/Post and this response cannot be sent, this should be handled as follows:

- Retry once after 2 seconds.
- Check in DNS-SD if the endpoint is still available or has changed.
- Retry again for 3 times every 2 seconds.
- Send a log message to the System Documentation on each state change in previous steps.

2.2.6 Service directory

[E-100] All services must be published using DNS-SD on mDNS.

[E-101] Applications must only publish a Service interface only when the HTTP communication can be functionally processed, i.e. when the application is up and running (not before).

[E-102] Applications must remove the publication for each Service interface in the service directory when the application is stopped.

2.2.7 Data

[E-110] Application which store data internally must, on start-up, verify this data for completeness and correctness. This data status should be used to determine which functionality can be provided and which service interfaces can be published.

Determining this 'degraded mode' is part of the design and should be agreed with GVB.

[E-111] If an application detects that stored data is incomplete or incorrect, it must send a log message to the System Documentation service conform [4].

2.2.8 Information exchange

Communication between services uses HTTP and XML for which XML-namespacing is used.

Information exchange can be synchronous (HTTP-GET) and asynchronous (via publish-subscribe).

Namespacing

Services use XML-namespacing. This is necessary to avoid conflicts in the (future) canonical datamodel.

The following examples are all parsed to field value Car.Colour=Blue:

Namespace example A (no namespace):

```
<ServiceDelivery>
  <Car>
    <Colour>Blue</Colour>
  </Car>
</ServiceDelivery>
```



Namespace example B (single namespace definition):

```
<cs:ServiceDelivery xmlns:cs="gvb.nl/carservice">
  <cs:Car>
    <cs:Colour>Blue</cs:Colour>
  </cs:Car>
</cs:ServiceDelivery>
```

Namespace example C (multiple namespace definitions):

```
<cs:ServiceDelivery xmlns:cs="gvb.nl/carservice" xmlns:gen="gvb.nl/generic_definitions">
  <cs:Car>
    <gen:Colour>Blue</gen:Colour>
  </cs:Car>
</cs:ServiceDelivery>
```

[E-120] Applications must support dynamic Namespace prefixes. Replacing a service by a version with different Namespace(s) should be possible without impact to consumers.



2.2.9 Platform and development

[E-130] Applications must support running on Windows 10 IoT Enterprise.

[E-131] Applications must, as far as possible, be portable to recent Linux distributions.

[E-132] Applications be developed in Java. Versions must be determined and agreed with GVB.

[E-133] Applications must support multithreading.

Performance and response times

The typical configuration of a GIVA Generic Computer is:

- Intel Core i7 processor
- Minimum 8GByte memory
- Minimum 32Gbyte storage
- Windows 10 IoT Enterprise

[req] The maximum delay between receiving information from a service, processing it, and providing updates on the service interfaces is 50ms.

Applications and services in vehicles can be separated in 3 categories based on resource usage.

Examples are:

- | | |
|------------|--|
| Category 1 | Applications with multiple functions, operations and interfaces.
AVMS application, RIV application. |
| Category 2 | Applications with limited functionality and high data frequency.
Vehicle Information service |
| Category 3 | Applications with minimal functionality, mainly translating data
GNSSlocation service, Netex service. |

[E-140] Category 1 applications can use the following maximum resources, averaged over 1 minute, during normal operation on the platform described above:

- 20% CPU
- 512Mbyte program memory, exc. data cache
- 5 Gbyte storage

Exceeding these figures is only allowed after GVB agreement.

[E-141] Category 2 applications can use the following maximum resources, averaged over 1 minute, during normal operation on the platform described above:

- 10% CPU
- 256Mbyte program memory, exc. data cache
- 1 Gbyte storage

Exceeding these figures is only allowed after GVB agreement.

[E-142] Category 3 applications can use the following maximum resources, averaged over 1 minute, during normal operation on the platform described above:

- 5% CPU



- 100Mbyte program memory, exc. data cache
- 500 Mbyte storage

Exceeding these figures is only allowed after GVB agreement.

3 GIVA peripheral devices

GIVA peripheral devices are physical devices, using information from GIVA services for displaying information etc.

3.1 Physical requirements

[E-200] Peripheral devices will be installed in closed compartments, only accessible to authorised (maintenance) personnel, except for devices for interaction with passengers/public and drivers/conductors.

[E-201] Devices must be installed so that LEDs and connectors can be reached/viewed without removing or uninstalling any equipment.

[E-202] Network interfaces will be implemented using M12 D-coded connectors for 100Mbit/s and M12 X-coded connectors for 1Gbit/s connections.

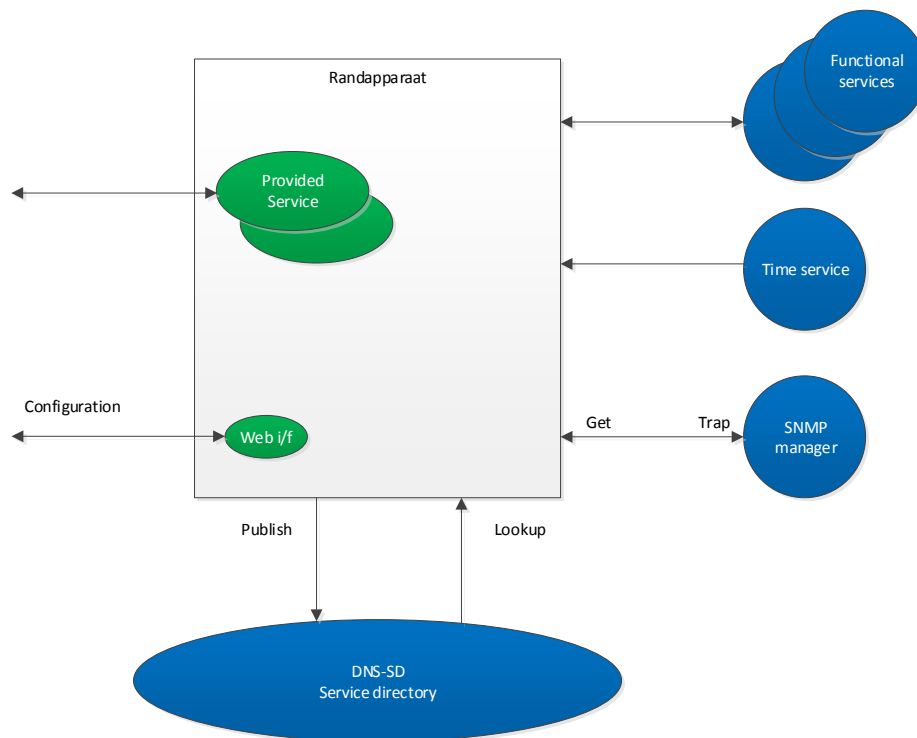
[E-203] Devices are preferably powered using Power Over Ethernet (PoE) to minimize cabling.

3.2 Functional requirements

Peripheral devices only use part of the GIVA platform, for:

- Service discovery
For registering the supplied services and discovering the services, needed to provide the required functionality.
- SNMP monitoring
SNMP provides a standard mechanism to monitor devices.
- Time synchronisation

The figure below gives a schematic overview.



3.2.1 Service discovery

All requirements from paragraph 2.2.1 are applicable.

3.2.2 Monitoring

[E-210] Each device will perform automatic diagnostics upon start-up and periodically during operation.

[E-220] Devices must support SNMP monitoring, both using SNMP traps and SNMP query/response.

[E-221] Devices must discover the location of the SNMP trap receiver in the Service Directory.

[E-222] Devices must publish the location of its SNMP client in the Service Directory.

[E-223] Devices must support SNMP version 3.

[E-224] Devices must provide the following status information through SNMP:

- Software and/or equipment versions
- Operational state information (e.g. start/init/operational/stop)
- Discovery state information for all GIVA services connections (e.g. found, subscribed, failed, ok, unsubscribed)
- Connection state information for all GIVA services connections (e.g. ok, data error).
- Internal failures and warnings



3.2.3 Device-specific web interface

All requirements from paragraph 2.2.3 are applicable.

3.2.4 Starting and stopping

Devices are switched off according a fixed pattern; by design, directly switching off power from an operational condition is avoided, however devices must be resistant to this situation.

[E-240] Devices must be resistant to spontaneous power cuts without any advance notice.

[E-241] The operational status of the vehicle (active, standby, off, etc.) is available through the Vehicle Information Service (element *Power state*). Devices can use this status for controlled power down.

Rationale: Physical power is only switched off, a certain time *after* the Power state is changed, to allow devices to properly shut down.

3.2.5 Service interface consumers

[E-250] Service interface consumers must validate all incoming messages to an XML Schema Definition (XSD) for the concerning interface.

[E-251] Service interface consumers must handle corrupt data and/or data which does not comply to the XML Schema Definition (XSD), without any impact on its normal operation.

[E-252] Service interface consumers must, upon receiving corrupt data and/or data which does not comply to the XML Schema Definition (XSD), report this status via SNMP. As soon as correct data is received again, the SNMP status must be updated to indicate the end of the error situation.

Service interfaces - Subscription

[E-253] Service interface consumers must be able to handle spontaneous subscription response with status false (unsubscription response) from a subscribed service.

[E-254] If a Service interface subscriber does not receive a message within one minute (60 seconds), it should handle this as follows:

- Unsubscribe at the concerning Service interface
- Check in DNS-SD if the endpoint is still available or has changed.
- Subscribe.
- Repeat the previous 2 steps every 2 seconds.
- Update the concerning SNMP connection status on each state change in previous steps.

Note: This is a keep-alive mechanism to verify if subscriptions are still active.



[E-255] Service interface consumers may not periodically subscribe to a Service interface if a subscription is already active.

3.2.6 Information exchange

All requirements from paragraph 2.2.8 are applicable.

3.2.7 Time synchronisation

[E-260] Devices must synchronise their system time to the Time service; the location of the Time service is available in the Service directory.

4 Appendix – best practices

To support generic behaviour between services and clients in the system GVB has set up a best practices for consuming and supplying services in GIVA.

Consuming GIVA services.

Service discovery

To consume service, this service *must* be resolved using DNS-SD, because IP-addresses and port numbers are dynamic and differ per vehicle.

To make use of DNS-SD it's recommended to use a proven DNS-SD implementation. Avahi is a C-implementation with good DNS-SD support. Avahi is brought under the LGPLv2.1 license model, which is friendly for reuse by suppliers. Another implementation which is already successfully used in GIVA is jmDNS for Java-based applications, this implementation is brought under Apache 2.0 license. Apple Inc. has made an implementation available under the name Bonjour. It's supported for Windows, Linux and MacOS. The usage of Bonjour is regulated under a proprietary license model of Apple Inc. If Bonjour is used, correct fulfilment of this license model is a responsibility of the supplier of the application. GVB does not accept any liability resulting from the usage of this middleware. When (re)connecting to a service, always use the latest updated information in the DNS-SD record. It's possible the endpoint has changed since the last connection attempt.

XML parsing

To parse XML-messages it's recommended to use a proven XML parser. Namespacing is used by GIVA services and this should be taken into account in the selection and usage of an XML-parser. XML-parser come in different shapes and forms, roughly a separation can be made between full blown xml parse libraries and low footprint parsers. Namespacing is mostly supported by full blown parse libraries. Low footprint parsers will most likely not recognize namespaces and consider it part of the element name (e.g. *avms:JourneyId* is parsed as string *avms:JourneyId* instead of string *JourneyId* in the namespace of *avms*).

Suppliers of embedded devices will most likely choose for low footprint parser, which have their limitations considering XML-namespacing. Quoting the documentation of the popular TinyXML¹:

Further, TinyXML has no facility for handling XML namespaces. Qualified element or attribute names retain their prefixes, as TinyXML makes no effort to match the prefixes with namespaces.

If a low footprint XML-parser is used (like tinyXML) instead of Xerces within GIVA it is allowed to pre-process the message and remove the namespace definitions. However the namespace prefix is not fixed and might differ. Therefor the pre-processor should support dynamic namespace prefixes.

¹ <http://www.wikiwand.com/en/TinyXML>