

PUBLICLY
AVAILABLE
SPECIFICATION

**ISO/PAS
22720**

First edition
2005-12-01

ASAM Open Data Services 5.0

ASAM Services de Données Ouvertes 5.0



Reference number
ISO/PAS 22720:2005(E)

AVA Documents
Document Sharing Hub
© ISO 2005

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO 2005

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

In other circumstances, particularly when there is an urgent market requirement for such documents, a technical committee may decide to publish other types of normative document:

- an ISO Publicly Available Specification (ISO/PAS) represents an agreement between technical experts in an ISO working group and is accepted for publication if it is approved by more than 50 % of the members of the parent committee casting a vote;
- an ISO Technical Specification (ISO/TS) represents an agreement between the members of a technical committee and is accepted for publication if it is approved by 2/3 of the members of the committee casting a vote.

An ISO/PAS or ISO/TS is reviewed after three years in order to decide whether it will be confirmed for a further three years, revised to become an International Standard, or withdrawn. If the ISO/PAS or ISO/TS is confirmed, it is reviewed again after a further three years, at which time it must either be transformed into an International Standard or be withdrawn.

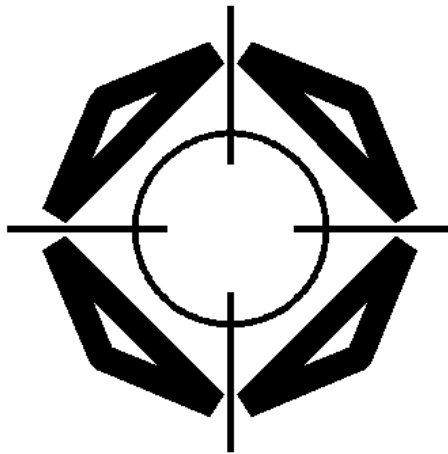
Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO/PAS 22720 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC 4, *Industrial data*, based on a description of the Version 5.0 prepared by the Association for Standardization of Automation and Measuring Systems — Open Data Services.

ASAM ODS VERSION 5.0

ISO-PAS

CHAPTER 1 INTRODUCTION



**Association for Standardization of
Automation and Measuring Systems**

Dated: 30.09.2004

© ASAM e.V.



Status of Document

Reference:	ASAM ODS Version 5.0 Introduction
Date:	30.09.2004
Author:	Rainer Bartz
Type:	Information
Doc-ID:	ASAM_ODS_50_CH01_Introduction.PDF
Status:	Release

Note: ASAM ODS has invoked a Formal Technical Review (FTR) process which intends to continuously improve the quality and timeliness of its specifications. Whenever an error is identified or a question arises from this specification, a corresponding note should be sent to ASAM (odsfr@asam.net) to make sure this issue will be addressed within the next review cycle.

Disclaimer of Warranty

Although this document was created with the utmost care it cannot be guaranteed that it is completely free of errors or inconsistencies.

ASAM e.V. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any expressed or implied warranties of merchantability or fitness for any particular purpose. Neither ASAM nor the author(s) therefore accept any liability for damages or other consequences that arise from the use of this document.

ASAM e.V. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.



Contents

1	INTRODUCTION TO ASAM ODS	1-3
1.1	GOALS AND BENEFITS OF ASAM ODS	1-4
1.1.1	POSITIONING OF ASAM ODS WITHIN ASAM	1-4
1.1.2	DRAWBACKS OF CURRENT SYSTEMS	1-6
1.1.3	BENEFITS OF ASAM ODS	1-7
1.2	TECHNICAL APPROACH	1-8
1.2.1	THE DATA MODEL	1-8
1.2.2	THE INTERFACES	1-10
1.2.3	THE PHYSICAL STORAGE	1-11
1.2.4	THE ASAM TRANSPORT FORMATS (ATF)	1-12
1.2.5	APPLICATION MODELS	1-12
1.3	IMPACT ON PRODUCTS	1-14
1.4	TECHNOLOGICAL LEVEL FOR IMPLEMENTATION	1-15
1.5	THE ODS DATA MODEL – A DEEPER INSIGHT	1-16
1.5.1	THE BASE MODEL	1-16
1.5.2	THE APPLICATION MODEL	1-19
1.5.3	THE INSTANCES	1-21
1.6	THE ODS APPLICATION PROGRAMMING INTERFACE (API) - A DEEPER INSIGHT...	1-23
1.6.1	THE OBJECT-ORIENTED API (OO-API)	1-25
1.6.2	THE PROCEDURAL API (RPC-API)	1-30
1.7	THE PHYSICAL DATA STORAGE - A DEEPER INSIGHT	1-32
1.8	THE ASAM TRANSPORT FORMAT (ATF) - A DEEPER INSIGHT	1-36
1.8.1	ATF/CLA	1-37
1.8.2	ATF/XML	1-38
1.9	APPLICATION MODELS - A DEEPER INSIGHT	1-39
1.9.1	THE NVH APPLICATION MODEL	1-39
1.9.2	THE CALIBRATION APPLICATION MODEL	1-39
1.9.3	THE VSIM APPLICATION MODEL	1-39
1.10	STATUS AND FUTURE STEPS	1-40
1.10.1	STATUS	1-40
1.10.2	FUTURE STEPS	1-40
1.11	REVISION HISTORY	1-41



Scope

This document is a brief description of ASAM ODS Version 5.0.

Intended Audience

This document is intended for people interested in ASAM ODS Version 5.0. It shall be used as a brief overview of the features of ASAM ODS Version 5.0.

This document is part of a series of documents referring to ASAM ODS Version 5.0 and must not be used as a stand-alone document. The documents referenced below build the technical reference of ASAM ODS Version 5.0 as a whole. They may be requested from the ASAM e.V. at www.asam.net.

ASAM ODS Specification

The following chapters build the technical reference for ASAM ODS Version 5.0:

- ASAM ODS Version 5.0, Chapter 1, Introduction
- ASAM ODS Version 5.0, Chapter 2, Architecture
- ASAM ODS Version 5.0, Chapter 3, Physical Storage (1.1)
- ASAM ODS Version 5.0, Chapter 4, Base Model (28)
- ASAM ODS Version 5.0, Chapter 5, ATF/CLA (1.4.1)
- ASAM ODS Version 5.0, Chapter 6, ATF/XML (1.0)
- ASAM ODS Version 5.0, Chapter 7: N/A ('Security' moved to Chapter 2)
- ASAM ODS Version 5.0, Chapter 8, MIME Types and External References (1.0)
- ASAM ODS Version 5.0, Chapter 9, RPC-API (3.2.1)
- ASAM ODS Version 5.0, Chapter 10, OO-API (5.0)
- ASAM ODS Version 5.0, Chapter 11, NVH Model (1.3)
- ASAM ODS Version 5.0, Chapter 12, Calibration Model (1.0)

Normative References

- ISO 10303-11: STEP EXPRESS
- Object Management Group (OMG): www.omg.org
- Common Object Request Broker Architecture (CORBA): www.corba.org
- IEEE 754: IEEE Standard for Binary Floating-Point Arithmetic, 1985
- ISO 8601: Date Formats
- Extensible Markup Language (XML): www.w3.org/xml



1 INTRODUCTION TO ASAM ODS

The rapid progress in hard- and software leads to storage of data in many different data base systems as well as under different hardware and/or server generations – not only within the automotive industry, but also within the supplying industry.

During development and production of vehicles, a huge mass of data is produced. Today, data are stored within the automotive industry in a standardised format specified by the ASAM ODS workgroup. ASAM stands for „Association for Standardisation of Automation and Measuring Systems“, and ODS stands for „Open Data Services“.

The ASAM ODS standard has the fundamental quality of storing data with an architecture-independent method. This leads to great advantages when exchanging data between different sources and possible prospective customers.

This document shall provide an overview of the goals of ASAM ODS as well as the technical approaches made to achieve these goals.

It is intended for readers with some technical background that want to get an impression on what ASAM ODS really standardizes and how the standards work. Readers may get an impression on what it means to implement these standards and what real benefit they may draw from using the standards.

Furthermore this document may be a starting point for implementers, before they dive into the detail standards documentation itself.

This chapter must **NOT** be seen as a specification; it is an introduction with some details to provide an overview to the interested reader.

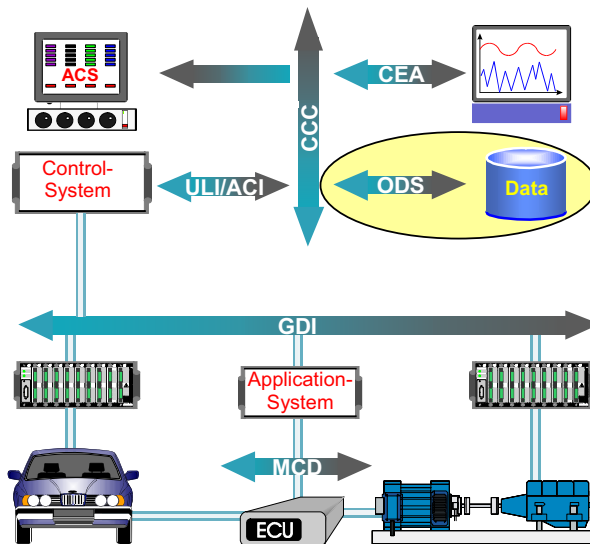


1.1 GOALS AND BENEFITS OF ASAM ODS

1.1.1 POSITIONING OF ASAM ODS WITHIN ASAM

The overall ASAM standards structure is shown in the figure, illustrating components which typically can be found in an automotive testing environment, and how these components use ASAM interfaces to communicate and interact with each other.

ASAM ODS is that part of the ASAM standards which focuses on persistent storage and retrieval of data.



➤ ASAM standards in an example testing environment ◀

ASAM ODS describes service interfaces. They can be used by any component of the testing environment to store its data and/or retrieve data required for proper operation.

Components using these interfaces are typically

- data acquisition systems, collecting data from a vehicle, an engine, etc.
- test control systems, used for running test procedures
- optimization tools, looking for an optimum set of calibration parameters
- analysis and reporting tools, presenting data to engineers and decision makers
- evaluation tools, supporting research and development tasks

Data stored and retrieved are typically

- test procedure configurations of a test control system,
- descriptive data of the test equipment and of the unit under test
- data measured during a test of an engine on a test bench, a vehicle in the wind tunnel, etc.
- data produced during a simulation run



- data resulting from test evaluation tools
- calibration data for engine or vehicle parts

The regular way to access ASAM ODS interfaces is, as the figure shows, through ASAM CCC (this is true for ASAM GDI/ACI and ASAM CEA as well).

ASAM CCC has proposed a server access based on the OMG-specified CORBA standard which requires object oriented system design.

CORBA does not specify what kind of services an interface provides; this is up to the specific service provider (in this case: ASAM ODS). Instead, CORBA specifies how an interface can be identified, found and connected to by any component that intends to use it. Thereby CORBA cares for a transparent network transfer, if required.

CORBA is an open standard and is wide-spread in all kinds of large-scale industrial and business processes. ASAM ODS specified an object oriented (OO-)API which can be implemented with the CORBA approach; however the defined interfaces are not restricted to CORBA and it may happen in the future that servers based on new technologies come up providing the same interfaces.

Because the first ASAM ODS interface specification used RPC as communication method and came up before ASAM CCC started its work, a lot of implementations still base on the RPC-API. For compatibility reasons the RPC-API will still be supported by ASAM ODS in the future, though the object oriented OO-API will be the main focus.

Since the access to a persistent data store typically requires network transfer, ASAM ODS is not generally optimized for realtime performance. However, ASAM ODS does not prevent realtime operation; time-related behavior depends on the design/implementation decision of a specific ODS-server as well as on the Object Request Broker (the ORB) used.

ASAM ODS specifies interfaces; the way data is finally stored persistently is not exclusively defined. While using some kind of object oriented or relational data base (with a common database model) might be reasonable, that decision is up to the ODS-server implementation.

Although ASAM ODS is the data storage/retrieval oriented part within the ASAM set of standards, it does not claim exclusive rights to specify file or database formats. Other ASAM standards may do so as well, if necessary. E.g. it may be required to store configuration or measurement data of a control system locally during operation. Or it may be required to use other internationally accepted data formats to exchange data with regulatory institutions (as e.g. is required for ASAM MCD aspects).



1.1.2 DRAWBACKS OF CURRENT SYSTEMS

Current systems in test, evaluation, and simulation environments within the automotive industry have in most cases their own proprietary formats to store data. These formats usually are very different from each other regarding the description of the configuration (unit under test, test sequence, test equipment, etc.) as well as the way results are stored (database, binary files, etc.).

View of the users:

Installations have grown during the past decades. Providers of the early days were replaced by others, each new one bringing in a new storage philosophy and another incompatible format. Accessing data from such a diversity of systems results in the need for a large variety of interface adaptors and converters - most of them individually developed or extended for each new project. The increasing complexity of the underlying systems causes expensive, specialized, isolated, and thus increasingly uneconomical solutions.

This is contradictory to the general desire within the automotive industry to develop and use easily manageable applications, which provide valuable information to other departments and systems within the company and to suppliers.

Additionally, even if alternative providers offer technically advanced systems, they often cannot be taken into account because of the amount of follow-up work required to link the already existing systems to them.

View of the providers:

Since each customer has built up a specific variety of systems, and interaction with these systems is usually strongly required with each new product being introduced, much of the development work of a new product is spent to become compatible. And not only the amount of work spent but also the knowledge required (and mostly gained through experience) is a critical issue. Thus the capability to connect into an existing environment often depends on the availability of specific people.

Moreover different customers require a different interaction scenario. Product releases become customer specific, version control becomes a major job.

New companies trying to contribute new ideas and solutions hardly have a chance to place any product; either they invest a lot into the connectivity issues or they will always play a minor role. This inhibits innovation in the automotive industry and finally jeopardizes competitiveness.

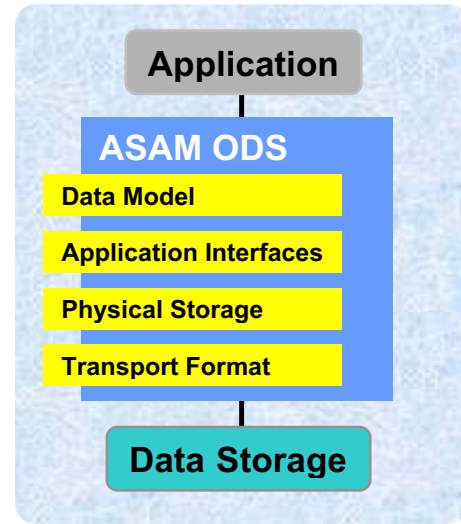


1.1.3 BENEFITS OF ASAM ODS

The main objectives for a standardization of data access interfaces are to reduce costs and risks within projects, and to provide a reliable basis for implementations in the area of data storage and data usage. Using standardized interfaces and common structures minimizes the efforts for the system integration within the heterogeneous environments discussed above and makes it much easier to exchange data.

To overcome the mentioned problems ASAM ODS provides:

- A common data model (base model) for the unambiguous and complete storage of data. This base model can be viewed as a rough data classification, thereby adding semantics to the data, which finally allows different systems to interpret same data in the same way. The data model covers the needs of a multitude of application areas and is adaptable to individual requirements of a specific system or even project by building a so called 'application model' from the base model.
- Interfaces (Application Programmer's Interfaces, APIs) to access data of ASAM-compatible systems and tools in a standardized way. Interfaces (APIs) to create and access a self-explanatory description (meta-information) of the actual application model. This allows systems to operate on very different ASAM ODS data.
- A database model for the (wide-spread) relational databases. This specifies physical storage of the data,. It also allows to exchange database files between systems with the same DBMS. Finally (in case an appropriate ODS server application is not available) it provides easy access through SQL commands even on different platforms, and between different DBMSs.
- A standardized, easy to use, text-based exchange format (ASAM Transport Format, ATF) in order to exchange ASAM ODS data (including its meta-information) between different systems and different platforms. The ASAM ODS Version 5.0 now supports both, the classic ATF (ATF/CLA) as well as the XML based ATF/XML.
- A set of application models that reflect typical scenarios for the use of ASAM ODS and that easily allow a mapping between data originating from different companies.



Another benefit of such a standardization is its impact on product quality. The standards specified will allow to measure products' implementations. Certification procedures are defined, another way to check the interoperability of products is to do crosstests, which were already undergone with very good results. Both approaches shall lead to product conformity with the standards.

Finally ASAM ODS provides the opportunity to integrate data of the whole lifecycle of automotive products. Though the kind of relevant information in the areas research, development, production, and after-sales are very different from each other, ODS allows to store them (each with their specific application models) and retrieve them, thereby keeping the meaning of the data items. And the ODS interfaces allow tools to combine information from each of those areas, analyze dependencies, generate overall reports etc..



1.2 TECHNICAL APPROACH

ASAM ODS standardizes data access regarding five important aspects:

- The data model
- The interfaces
- The physical storage
- The transport formats
- Application Models

1.2.1 THE DATA MODEL

The first and most important aspect of ASAM ODS is its **data model**.

Typically there are different levels to exchange a data item.

- Two systems or software packages might not be able to exchange this data item at all. This is the case if e.g. system A expects the data item to be represented as a 4-byte floating point number while system B expects it to be represented by a 2-byte integer value.
The drawback is obvious: no automatic data exchange can be set up without changing either of the systems (that is: its software).
- Two systems or software packages may be able to exchange this data item as a number because the data representation of the item is the same on both systems. This capability often is the result of some standardization effort, specifying universal data formats. An example is the wide-spread CSV (comma separated values) format, where data items are represented by an ASCII string, and consecutive data items are separated from each other by a comma. System A may produce such a CSV file and system B may be able to read it and know about the numbers contained in it.
The drawback is still obvious: without additional knowledge about the meaning of the data item its value is quite useless.
- Similarly two systems or software packages may be able to exchange this data item as a number with a name e.g. because a common database is used. In this case system A stores the value of the data item in some place in a table, using a row and/or column name; this is typically done through some database interfaces. System B may now use the same database interfaces, identify the data item by its name and/or location in the table and thus get back its value.
The drawback is not very obvious; it seems that the data item is fully described by (i) its value and (ii) its name. However, its value is only useful if the retrieving system B knows about the meaning of the data item's name. This requires additional conventions on naming of data items. And questions like "What unit belongs to the value?", "Is this the most recent of a set of available values for the data item?" etc. still remain open.
- Some data exchange solutions overcome this drawback by specifying everything. Examples can be found looking at some serial interface protocol specifications or at some quite fixed database models. Though this provides a maximum of information provision, such solutions proved to be very inflexible and are tailored to the needs of one specific task.



ASAM ODS headed for an optimum compromise.

On the one hand the specifications shall be applicable to a variety of application areas. Data gathered from small vehicle parts shall be accessible in the same way as those of complete vehicles. Data taken in overseas shall be available together with on-site data (regardless of the language, unit schemes, etc.). Data from research projects as well as from production processes or even after-sales information shall be exchangeable. All systems involved in any of those steps shall be able to store and retrieve data based on the ASAM ODS data model. And, since large amounts of data already are existing all over the companies, ASAM ODS should provide some means to integrate them into the new approach.

On the other hand the information stored shall still be valuable and contain enough meaning to allow some automatic retrieval and data analysis. There should be a relation between a data value (the number) and a corresponding unit in order to know about the real physical value. There should be a relation between the data item and the vehicle or part where it was taken, and with what equipment it has been taken, and who has taken it, and so on.

This finally led to the current data model of ASAM ODS. It distinguishes between a so called 'base model' and an 'application model'.

The base model is a set of defined base elements and a set of base relations between them.

Each base element represents a type of information. E.g. 'AoUnit' is the base element that represents information on a physical unit (like Newton, Kelvin,...), 'AoMeasurementQuantity' is the base element that represents information on a measured physical quantity (like force, temperature,...), and so on.

Each base relation represents a link with a specific meaning between two such base elements. E.g. AoMeasurementQuantity and AoUnit are linked together and the relation tells which of all possible units is the current unit of that quantity.

The base model is unique for all kinds of applications that use ASAM ODS.

The application model is application specific. A test system for wind tube tests may have a different application model than a system running engine endurance tests.

The application model specifies which elements are really in use by the application. For example, an engine test shall measure a temperature, a pressure, and a force and will use the SI units Kelvin, Pascal, and Newton. These six instances are quantities going to be measured (thus belong to the base element AoMeasurementQuantity) or units (thus belong to the base element AoUnit). Each of them is an individual application element (derived from the corresponding base element), and there is a relation between each such quantity used and an appropriate unit.

Though each application may have its own application model, all application elements in that model know of which base element type they are. And any software accessing such an application element knows more about it than just name and value: it knows the type of information this element carries, it knows in which unit its values currently are expressed and so on.

The data model is explained in more detail in section 1.5.



1.2.2 THE INTERFACES

The second aspect of ASAM ODS is its set of **interfaces**.

It is obvious that storing and retrieving information needs some kind of physical storage medium (like a disk, a tape, etc.). Again, there are different levels to store/retrieve information.

Usually there is an operating system which provides a basic interface to read and write single data items to/from files on the storage medium. This is a quite low level of interface, which requires that each single byte within that file is uniquely specified. Otherwise, though system A could write its information to file in some way, system B would probably not be able to read it back. Besides the huge amount of specification work required, there is still a big chance that system implementers will misunderstand or misinterpret the specifications and produce incompatible files, especially if there are dependencies between data items in the files.

A more convenient way is to use a DBMS (database management system) and to store the information in a database. Today, relational databases are quite wide-spread, and access to the contained information can be gained by using the standardized SQL (structured query language) interface provided by the DBMS. In this case applications don't have to care for the exact file representation of their data; they just use the SQL-commands to store and retrieve data items, and the DBMS takes care for that a data item stored by system A can be retrieved by system B.

Data in relational databases are organized in tables which may be related to each other. Thus specification work has to concentrate on specifying the tables and their relationships. ASAM ODS has done so (see the 'physical storage' description below).

However a drawback of a solely database oriented standard is that all preexisting data not contained in databases will not be accessible as ASAM ODS data unless they are converted/imported into a database.

Another drawback is that a DBMS will never know about ASAM ODS specific aspects and thus cannot guarantee consistency of the database contents regarding ASAM ODS criteria.

Additionally DBMSs typically require some amount of administrative work (and adequate personnel) and involve license fees. This may become a problem for some small scale application areas.

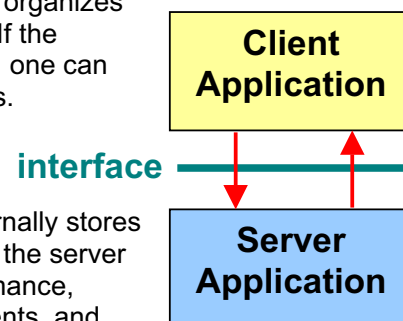
The most promising solution is to provide a specific interface standard that is closely coupled with the ASAM ODS data model specification. An interface separates a client application from a server application. The server application holds and organizes the data; the client application stores or retrieves the data. If the interface considers the ASAM ODS specific characteristics, one can expect an optimal co-operation between clients and servers.

This is the concept ASAM ODS has adopted.

The server provides interfaces to store or retrieve data.

The client may use these interfaces. The way a server internally stores the data persistently is completely up to him. The quality of the server implementation will influence its behavior regarding performance, overall data amount, simultaneous access from several clients, and others.

Due to the complete encapsulation of the data store, the server application may decide to use a relational or object oriented database for persistent storage. Alternatively, just a flat file storage system may be used, if this seems more appropriate for the intended application





area; clients will not recognize any differences in functionality. Moreover this encapsulation allows to build server applications which can access preexisting data (in whatever storage format) and present them to any client application as if they were proper ASAM ODS data.

Version 5.0 of the ASAM ODS standard includes two different types of interfaces:

- a procedural interface based on the RPC technology and further on referenced as the RPC-API.
- an object-oriented interface further on referenced as the OO-API. This is currently based on the CORBA architecture but may in the future be expanded to other technologies.

The RPC-API will still be supported by ASAM ODS in the future, though it currently is frozen; no further revisions of that API are expected to come up.

The interfaces are explained in more detail in section 1.6.

1.2.3 THE PHYSICAL STORAGE

The third aspect of ASAM ODS is the specification of a format for **physical storage**.

Originally several different physical storage formats were intended to be specified: one for (relational) database oriented storage, one for storage using a set of individual files ('flat-file'), one for a so-called 'mixed mode', where mass data reside in files while descriptive data are contained in a database.

Up to now ASAM ODS has specified how a relational database should internally be constructed to store information in ASAM ODS compatible way. This includes defining which tables must be set up, what information they have to keep, which columns are keys (and thus have to be unique) and so on.

It is not really necessary to specify a physical storage format, to come to a standardized data storage concept.

Moreover, the major benefits from the ASAM ODS standards can be achieved by just implementing the data model and the interfaces, without regarding any specifications on physical storage formats (products will still seamlessly interact).

However specification of such a physical storage format provides some benefits.

- First of all, if different server applications use the same DBMS on the same platform, and their data have to be exchanged, the database files may directly be copied. No time consuming export-import procedures are required to transfer the data from one server application to the other.
- Additionally regular SQL-based tools (browsers, analyzers, report generators) may be used to access the data, bypassing the server application. Such tools are available from a variety of vendors, are often easy to handle, not very expensive, and sometimes the existing staff is already familiar with their usage. Though there is no ASAM ODS specific interpretation of the data available, such tools may be sufficient (especially for simple tasks); in this case no ASAM ODS server application needs to be licensed and installed.
- Finally the ODS-server itself may be exchanged without loss of data or the need to export the data out of the original server and import them into the new one again.

The physical storage is explained in more detail in section 1.7.



1.2.4 THE ASAM TRANSPORT FORMATS (ATF)

The fourth aspect of ASAM ODS is the specification of a **transport format**, the so called 'ASAM Transport Format' **ATF**. It is intended to be usable on all available platforms and operating systems and shall allow data transfer between different systems without losing any information.

Since there is hardly a system or platform where plain ASCII text is not available as information representation, ASAM ODS decided to specify the ATF based on ASCII text. The specified syntax allows to carry all information of the data model.

ASAM ODS transport format files may become quite large due to their ASCII nature; therefore the specification allows to separate the complete information into several single ASCII files and moreover allows to place mass data into separate binary files whose internal structures then are described within the ASCII file.

It is expected that ASAM ODS server applications will allow to import ATF files and to export all or part of their information into ATF files. This applies also to ASAM ODS clients using the API to import and export ATF files.

The ATF will typically be used to transfer e.g. measured data from a stand-alone in-vehicle measurement system to the department's ODS server, or to transfer parts of an ODS server (or its whole content) from one platform to another, in case there is no CORBA-capable network connection available. Because it is ASCII-text, the ATF can also be used to manually analyze or modify specific data items.

Another benefit of the ATF is that it can easily be created by any application. There is no need to know about CORBA or RPC nor to have experience with software interfaces or databases. Thus it may be a first and easy step for any product to become ASAM ODS compatible.

Version 5.0 of the ASAM ODS standard includes two different transport formats:

- the first one specified is the so-called classic ASAM transport format (ATF/CLA). It has been used by servers and clients for years and is a stable and reliable way for ASCII based data exchange.
- with XML becoming a base language for describing data storage in a huge variety of application areas, ASAM ODS has now introduced the ATF/XML, an XML based definition of file contents that allows exchange of data between ASAM ODS compliant applications.

The ATF/CLA will still be supported by ASAM ODS in the future, though no further revisions of that ATF type are expected to come up.

The ATF is explained in more detail in section 1.8.

1.2.5 APPLICATION MODELS

The ASAM ODS Version 5.0 now also includes application models to allow implementers to easily implement application models which are very common. At the time of publishing this documentation, the following application models are available or under development:

- NVH (Noise, Harshness, Vibration)
- Calibration Data
- VSIM (Vehicle Safety Information Model)



INTRODUCTION To ASAM ODS

- Engine
- Emissions

Application models defined up to now are explained in more detail in section 1.9.



1.3 IMPACT ON PRODUCTS

ASAM ODS will influence already existing software-based products which perform measurement, simulation, and control tasks, or which organize and analyze data or create reports:

- Large automation, data acquisition, or analysis products will be extended to use the ODS interfaces to connect to an ODS server. They will store their configuration information as well as the test results using the ODS server.
- Data acquisition products with rather small functionality might show up with only an ATF file as output data; implementing the ASAM ODS interfaces to connect to a server may be too expensive for them.
- Simulation tools may be extended to not only perform simulation based on their internal stimulation processes but also based on already existing data from previous test; those may easily and uniquely accessed through the ODS interfaces.
- Calibration and optimization packages may combine actual measurement results of a data acquisition system with results taken in the past and stored in an ODS server (accessing them through the ODS interfaces). This may help reduce overall time amount for testing and may increase reusability of data.
- Report generators and presentation tools now can access data from any testing environment in a standardized way through the ODS interfaces and thus more easily. A comparison of results between tests from different facilities will become available independent from the equipment that produced the results.

ASAM ODS will provide opportunities for new products:

- ASAM ODS server applications will be developed. They may range from small Windows-NT based solutions for a limited number of similar test cells to large enterprise oriented systems which will be able to manage data from very different sources. Some may decide to use a relational database and exactly the specified physical storage format, others will base on their own proprietary format (using e.g. object oriented databases or just a set of 'flat files')
- Data browsers and editors may provide a quick overview over the contents of the ODS data store. Some may be capable to combine the data of several ODS servers into one view. Others may only be capable to work with ATF files.
- Connectors between existing universal packages like EXCEL[®], MATLAB[®], etc. and the ODS data store will be offered (e.g. through the use of VBA).
- Analysis tools and report generators may come up as CORBA or RPC (software) components; they are quite lightweight and focus on a specific task, but can be combined with other components within an adequate framework to build up an individual tool with exactly the functionality needed.



1.4 TECHNOLOGICAL LEVEL FOR IMPLEMENTATION

Besides good software development skills and the knowledge of the ASAM ODS specification some key technologies are required to understand and implement ASAM ODS:

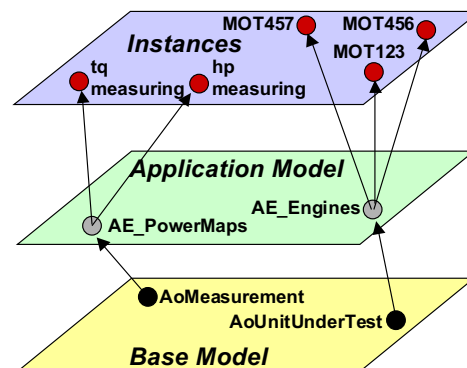
- The ASAM ODS data model is specified using the STEP EXPRESS syntax. Being familiar with this syntax is beneficial. Though there is a graphical illustration of the relationship between ODS elements, which can be understood quite easily, those diagrams don't contain all information. Experience with data modeling tools, entity relationship models etc. is helpful.
- The ASAM ODS interfaces use CORBA IDL or the RPC IDL as interface definition language. To understand the interfaces' functionality typically requires no extra knowledge.
To use them as a client requires a CORBA ORB implementation (only needed when using the OO-API based on CORBA; such an ORB is available from different suppliers), an ODS server (some are already available from ASAM members) and some experience with this specific way of working with remote objects and their local proxies.
To implement them as an ODS server is a much more demanding task. It requires experience with CORBA server implementations, or, in case of the RPC-Interface, with RPC server implementations.
- The currently specified physical storage is based on relational databases. An appropriate DBMS is required to use it. Additionally, some experience with administration of DBMSs and experience using SQL should be available.
- Implementing an import/export from/to an ATF/CLA file does not require anything specific. Implementing it with ATF/XML requires knowledge about the XML technology.



1.5 THE ODS DATA MODEL – A DEEPER INSIGHT

This section will give more detailed information on the first aspect of the ASAM ODS standard. However, to implement this standard will require to study chapters 2 and 4 of the ASAM ODS Version 5.0 documentation.

The data model of ASAM ODS distinguishes between a 'base model' and an 'application model'. Both are describing the structure of the data stored. Real values are finally stored in instances of application elements. The data model thus is a three-layer model, and an introductory example (without any further discussion) is shown in the figure. These subjects will be explained in this section.



1.5.1 THE BASE MODEL

The base model is completely defined by ASAM ODS. It consists of a set of base elements and a set of base relations between them.

Base elements are something like information classes. The table and the figure show the base elements currently specified, and how they are grouped together. The prefix Ao represents a shortcut for ASAM ODS.

Base relations are links between those base elements; each of these links carries a specific meaning.

AoTest, **AoSubTest**, and **AoMeasurement** are used to organize measurements and corresponding input/output data. These three elements allow applications to build up an organizational hierarchy for the tests performed and manage the test results.

AoSubMatrix and **AoLocalColumn** are the base elements where test results are stored. ASAM ODS assumes that result data can be arranged in tables (submatrices) where each column represents the values of a measurement quantity and each row the values of subsequent measurement points.

A measurement may contain several such submatrices; they may be combined to one large measurement matrix (which typically will have quite large empty areas; this is why storing the results is done through a set of submatrices rather than through one measurement matrix).

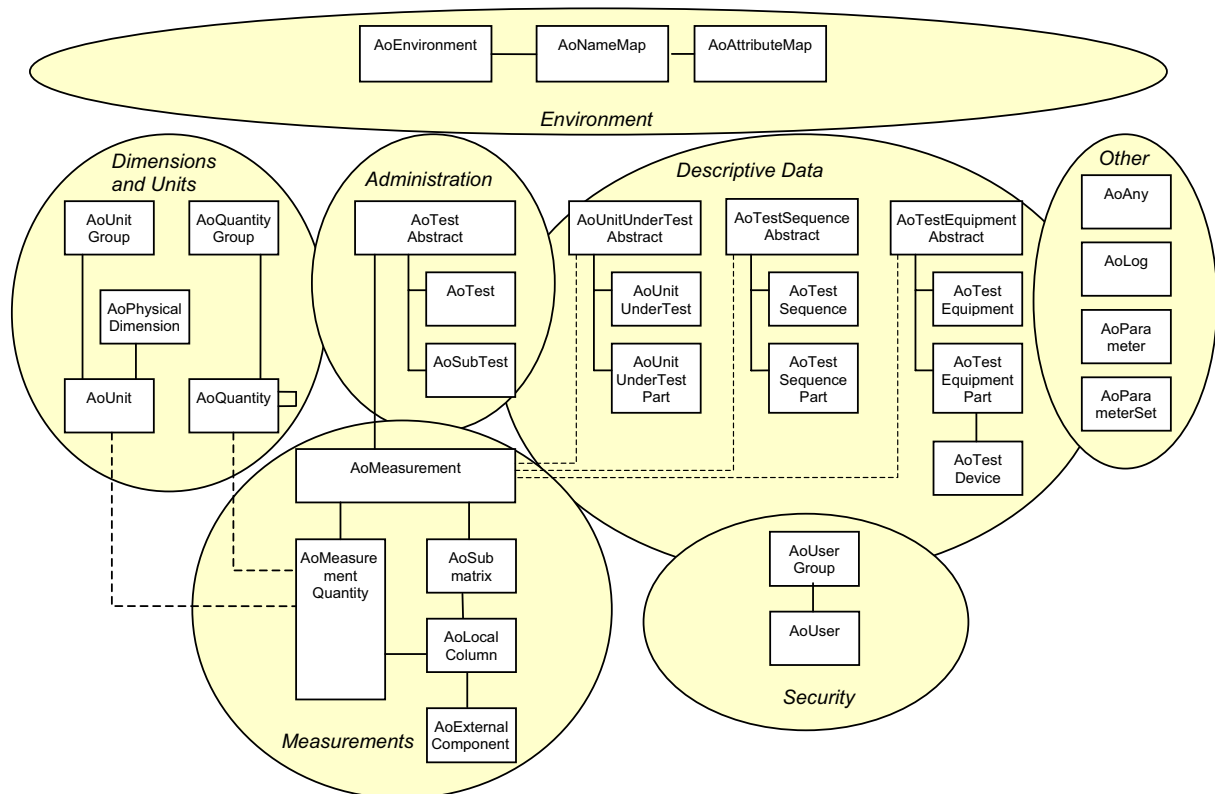
AoQuantity is used to keep information on a physical quantity that may be of relevance for any or all of the tests kept in the database. **AoQuantity** is linked to **AoUnit** which means that there is a default unit for each quantity.

Base Elements

- AoTest
- AoSubTest
- AoMeasurement
- AoSubmatrix
- AoLocalColumn
- AoMeasurementQuantity
- AoQuantity
- AoQuantityGroup
- AoUnit
- AoUnitGroup
- AoPhysicalDimension
- AoUnitUnderTest
- AoUnitUnderTestPart
- AoTestEquipment
- AoTestEquipmentPart
- AoTestDevice
- AoTestSequence
- AoTestSequencePart
- AoUser
- AoUserGroup
- AoLog
- AoParameter
- AoParameterSet
- AoNameMap
- AoAttributeMap
- AoAny



INTRODUCTION TO ASAM ODS



AoMeasurementQuantity represents those quantities that are used in a measurement (e.g. where test results are available for). **AoMeasurementQuantity** is linked to **AoQuantity** and **AoUnit**.

AoUnit and **AoPhysicalDimension** are used to keep information on the relationship between result values (plain numbers) and the corresponding units, thereby caring for correct result values (even in case the unit system is changed e.g. between english and metric).

The pairs {**AoUnitUnderTest**, **AoUnitUnderTestPart**}, {**AoTestEquipment**, **AoTestEquipmentPart**}, and {**AoTestSequence**, **AoTestSequencePart**} contain information on

- what has been tested (**UnitUnderTest**, UUT)
- with which equipment the test has been performed (**TestEquipment**, TE)
- which sequence of steps was processed during the test (**TestSequence**, TS)

The base relations within each of these pairs are specified by the base model in a way that real applications may build up a hierarchical tree; the root element is e.g. a UUT, which may have one or more UnitUnderTestParts which again may have one or more UnitUnderTestParts and so on.

AoUser and **AoUserGroup** are base elements used for security aspects.

As ASAM ODS is undergoing a continuous development, the ASAM ODS Version 5.0 incorporates the following new base elements: **AoLog**, **AoParameter**, **AoParameterSet**, **AoNameMap**, and **AoAttributeMap**.

AoLog is foreseen for logging purposes.



AoParameter and **AoParameterSet**: They can be used e.g. to avoid redundancy by storing information that is used by several elements only once. Alternatives would be to add application attributes or instance attributes, which would create a lot more data.

AoNameMap holds any number of alias names for an application element (e.g. for language versions). The list allows different language versions switched by the application software.

AoAttributeMap specifies any number of alias names for an application attribute..

AoAny, finally, is a base element that does not carry a specific meaning. It may be used to store all information that does not fit to the other base elements' meanings. However it should be used carefully; using any other base element to store information should be preferred.

Besides the specified base relations between the base elements, each base element has a well defined set of base attributes. These are characteristic values that describe the element in more detail. Base attributes may be mandatory or optional.

Each base element has at least the base attributes 'name' and 'ID' and may have attributes like 'version', 'version_date' and 'description'.

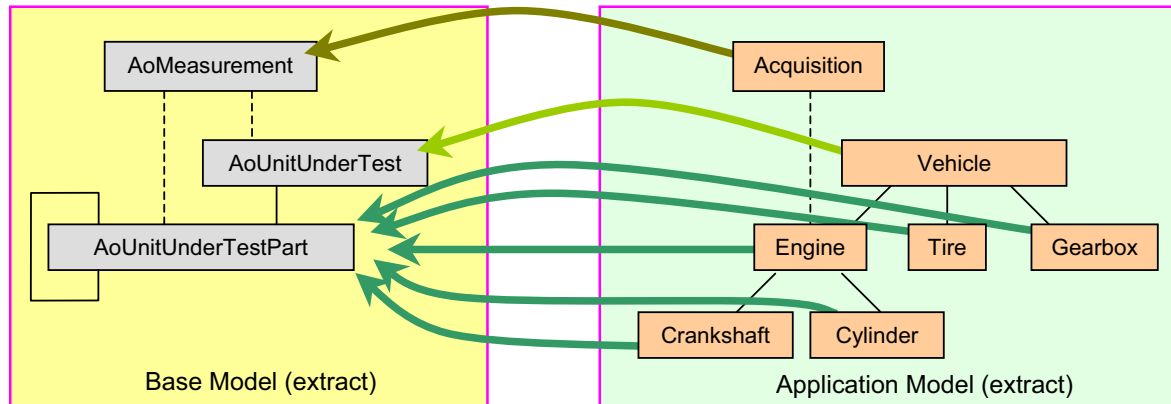
Some base elements have specific attributes: e.g. AoMeasurement is characterized by the attributes 'measurement_begin' and 'measurement_end', AoMeasurementQuantity has the additional attributes 'datatype', 'type_size', 'rank', and 'dimension' which are required to handle the data that are stored for this quantity.



1.5.2 THE APPLICATION MODEL

The application model must use the base elements to model the information that a real application intends to store or retrieve through ASAM ODS. Building an application model means specifying a set of application elements, each being related to one of the base elements, such that each information item to be stored can be placed into one application element.

The figure shows an extract from an example application model and the related part of the base model.



The base model extract shows the base elements AoMeasurement, AoUnitUnderTest, and AoUnitUnderTestPart. The lines between the base elements are the base relations.

The cardinality of the relations (which is not shown in the figure) is in this case

- [0,n] for all relations to/from AoMeasurement;
- an AoUnitUnderTest as well as an AoUnitUnderTestPart may have [0,n] AoUnitUnderTestParts
- an AoUnitUnderTestPart may belong to one or more AoUnitUnderTest(s) or AoUnitUnderTestPart(s).

The application model extract shown is one possible setup and in this example it is dedicated for engine testing. The designer of this application model has made following decisions:

- He called each measurement action an 'Acquisition' and thus has put an application element 'Acquisition' into his application model; it is of base type AoMeasurement.
- He decided, that though his main testing focus is on engines, some information on the vehicle, and other parts are relevant (e.g. for some post processing). Therefore he has put the application element 'Vehicle' as the root element of his test object description; it is of base type AoUnitUnderTest.
- He has decided that (at least for his processing needs) the vehicle consists of three different types of parts: the engine, the gearbox, and the tires. He thus put three application elements of type AoUnitUnderTestPart into his application model. Each of them is related to one vehicle.



- He has decided that (at least for his processing needs) the engine itself consists of two different types of parts: the crankshaft and the cylinder. He thus put two more application elements of type AoUnitUnderTestPart into his application model. Each of them is related to one engine.
- He has decided that an acquisition action should be related to the engine from which data are collected. The 'Acquisition' should know about which engine was mounted and the engine should know about which 'Acquisitions' have already been collected.
- These decisions led to the above displayed application model extract.

Besides the specified base relations between the base elements, which appear again in the application model as relations between application elements, the designer of an application model may introduce new relations between application elements (especially relations between application elements whose corresponding base elements are not related). Thus the whole set of application relations consists of those between elements that are already related in the base model and those that are added and extend the base model.

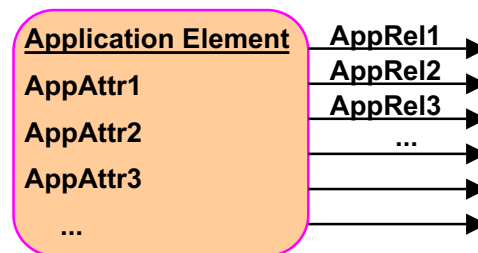
The same is true for application attributes: Each application element has a set of attributes that quantify its characteristics. The mandatory base attributes must also appear in each corresponding application element, the optional base attributes may appear in the application elements.

Besides these base attributes the designer of an application model may add application attributes that don't appear in the corresponding base element. This will usually be the case, since base attributes cannot be specific; they must be useful for the whole variety of possible application elements.

In the example shown above, the designer could have decided to add the application attribute 'weight' to the application element 'Vehicle', to add the attribute 'number_of_cylinders' to the element 'Engine', and to add the attributes 'bore', 'stroke', and 'spark_plug_type' to the element 'Cylinder' etc..

The figure summarizes the application element as the building block of the application model:

- Each application element corresponds to one base element ("is type of").
- Each application element has attributes that further quantify its characteristics; some are already specified in the corresponding base element, others are added.
- Each application element has application relations to other application elements; some are already specified in the base model as relations between the corresponding base elements, others are added.



With these considerations in mind one can state that there are several aspects within the ODS data model where meaning is transported with the data:

- Application elements get a meaning by the base element they correspond to.
- Application attributes get a meaning by the base attribute they eventually correspond to; even if they don't correspond to a base attribute, they contain some meaning because they belong to an application element with meaning.



- Application relations get a meaning by the base relations they eventually correspond to; even if they don't correspond to a base relation, they contain some meaning because they belong to an application element with meaning.

1.5.3 THE INSTANCES

Once an application model is designed, the structure of the data stored/retrieved is defined. Neither base model nor application model store any (measured or descriptive) values. They just provide 'meta-information' to know about the structure of the data store.

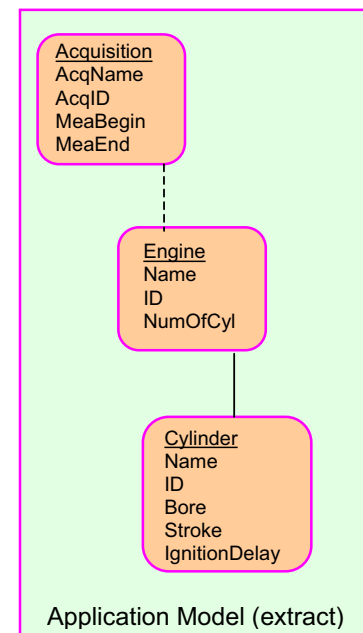
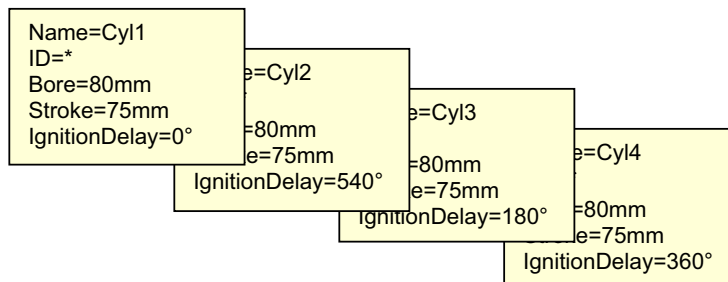
Whenever real values have to be stored, an instance element must first be created from an application element.

Instance elements can store values for each application attribute of the corresponding application element. Additionally new instance attributes may be added that do not appear at the application element.

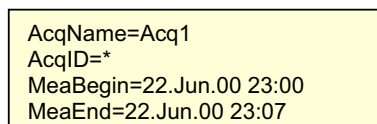
Instance elements can store relations (identifiers) to instances of other application elements; each application relation between two application elements AE1 and AE2 may result in one or more relations between instance elements of AE1 on one side and instance elements of AE2 on the other side. Additionally new relations between two instance elements may be added that do not have a counterpart between the corresponding application elements.

With the application model in the figure a measuring system may start to create instances as follows (the values for the ID attributes are set by the ODS server):

- one instance element of application element 'Engine':
- four instance elements of application element 'Cylinder':



- one instance element of application element 'Acquisition'





Further on, the measurement system will set a relation attribute of each of the cylinders to point to 'NewEngine', four relations of the engine to point to 'Cyl1', 'Cyl2', 'Cyl3', and 'Cyl4' respectively, one more relation of the engine to point to the 'Acq1' instance and finally one relation of the 'Acq1' instance to point to 'NewEngine'.

For each subsequent measurement that will be performed a new instance of the application element 'Acquisition' will be created, having a new name, a new (server-provided) ID and specific values for MeaBegin and MeaEnd.

(Actually, the measurement system will do a lot more things. For example, it will create instances of the application elements that finally store the sampled data. Those application elements are of base type 'AoSubmatrix' and 'AoLocalColumn' and were not contained in this small example. And it will create instances for measurement quantities, and units, and test equipment, ...)



1.6 THE ODS APPLICATION PROGRAMMING INTERFACE (API) - A DEEPER INSIGHT

This section will give more detailed information on the second aspect of the ASAM ODS standard. However, to implement this standard will require to study chapter 9 (resp. 10) of the ASAM ODS Version 5.0 documentation.

ASAM ODS provides two different types of interfaces:

- a procedural interface (RPC-API), which has been established as the first official ASAM ODS interface, and which is used by the majority of ASAM ODS servers and clients operating today.
- an object oriented interface (OO-API), which provides state-of-the-art access to data and more comfortable methods to work with the data. Currently available implementations base on the OMG's object model and the CORBA platform though other technologies may be supported in the future as well.

An ASAM ODS compatible server must implement either the full functionality of the OO-API or of the RPC-API. He may implement both. The product description of a server must clearly state which of the APIs (and what version of them) are implemented; otherwise he may not claim to be ASAM ODS compliant.

An ASAM ODS compatible client may implement a subset of either API. There are no minimum functionality requirements; however, what is implemented must be implemented completely according the API specifications. The product description of a client must clearly state which of the APIs (and what version of them) are used (and thus required from a server for interaction); otherwise he may not claim to be ASAM ODS compliant.

From time to time ASAM ODS offers the opportunity for cross tests. Several clients will be connected with servers and the behavior of them within a predefined scenario is investigated. The results are further discussed and may be used for verification and further development steps. Server and client implementers may register with ASAM for participation within such a cross test.

There is one definition file for each of the APIs, available from ASAM: The OO-API is defined in *ods.idl*, the RPC-API is defined in *aods.x*.

All modules and functions of the OO-API are explained in detail in chapter 10 (including purpose, calling sequence, return values, and example code) . All methods defined for the RPC-API are described in chapter 9.

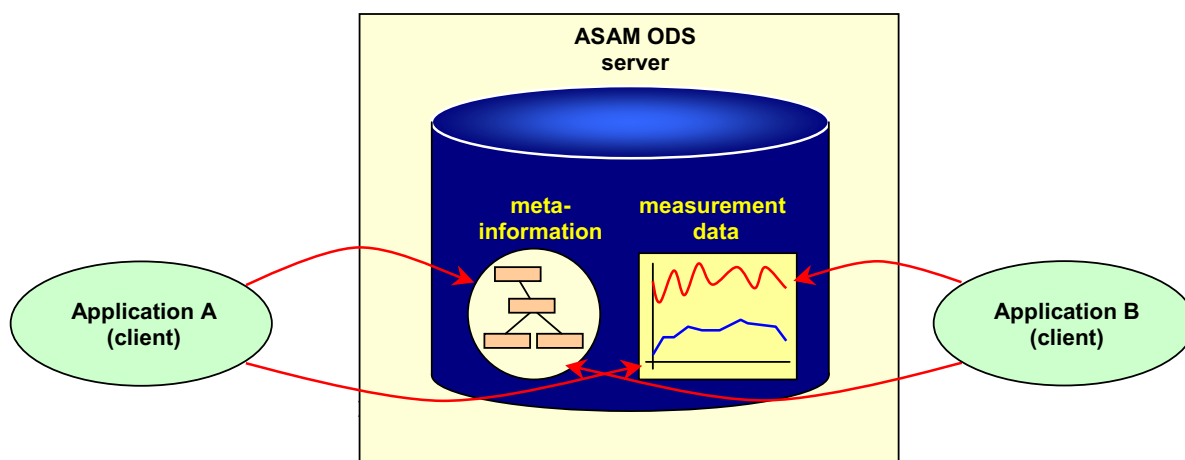
There are some differences between the functionality of the two APIs. The functionality of the RPC-API is a subset of that of the OO-API. The following list gives an overview on the major functionality aspects missing in the RPC-API:

- No methods to create/change the application model (read-only-access)
- AoSubmatrix and AoLocalColumn are handled through special methods (in the OO-API they can be accessed as normal application elements)



- Data types used for transporting sequences of application-attribute values (e.g. DS_LONG) are not supported (though sequences are supported within measurement data)
- No iterators on instances (queries return all data in one return-structure)
- No value matrix on the server (submatrices must be merged by the client)
- No transactions
- No locking

The ASAM ODS APIs separate client applications from the server application. The figure shows the principle of operations.



Within an ASAM-ODS server there are two major types of information:

- the data themselves (measurement data, descriptive data for test equipment, ...)
- the information about content and organization of those data (so-called meta-information), represented by the application model, which itself relies on the base model.

Both types of information may be accessed through client applications. Thereby the typical sequence of usage for a client that wants to read data from the server will be:

- connect to the server
- read the meta-information
- access the data themselves

When specifying an API for ASAM ODS several general and functional requirements have to be considered.

The major general requirements are:

- The API shall hide any network used to connect the clients with the server.



- The API shall take into account that some clients may want to exchange huge amounts of data with the server while other clients may want to access data items one by one. Both scenarios should not unnecessarily load neither the server nor the network.
- The API shall be available for all major platforms and programming languages. This allows both client and server implementations to decide on platform and operating environment independent from the API specification.
- The API shall allow to perform transactions. This means that from a sequence of actions using the API either all actions succeed or all actions may be withdrawn by the client (as if they were never initiated). This shall guarantee data consistency.

The major functional requirements were:

- The API should provide access to all information regarding the base model, the application model, and the instances.
- The base model is provided as information only; it cannot be modified through the interfaces.
Application model and instances must be available for read and write operation.
- The API shall care for that the rules defined for the ASAM-ODS data model are kept.
- The API shall provide means for access control. Some operations on the data should be restricted to clients with specific rights.
- The API shall provide a basic search and selection functionality to reduce network load.

The following sections present a short introduction to the APIs and should give a rough overview on the underlying principles and on how to start with an application. There is no way around the specification chapters 9 resp. 10 to correctly use the APIs.

1.6.1 THE OBJECT-ORIENTED API (OO-API)

The ASAM-ODS OO-API is actually a set of interfaces that are used to access data in an ASAM-ODS data storage. The OO-API has been specified considering the above general and functional requirements.

Investigations of an ASAM working group found that CORBA is the only non-proprietary distributed object-oriented transport mechanism that allows to define an API that fulfills the requirements listed above. CORBA is a standard specified by the OMG (Object Management Group), an organization created and supported by major IT companies. The basic idea of CORBA is to provide a transparent communication mechanism that resides on a set of systems and cares for the mediation of services between applications running on those systems. A kernel, the so-called ORB (Object Rquest Broker), is running on each of the systems. Whenever there is an application on a system that provides an interface (that is: provides means to instantiate an object and use the object's methods) it informs its ORB about this interface and will be regarded as a server application. Any application on the same or another system may now connect to this server as a client (either statically or dynamically by asking the ORB for available servers) and use this interface.

As a consequence, interacting objects residing on different platforms may communicate transparently over a network. The object implementations remain free of any transport-related code; transport is solely in the responsibility of the ORB.



ORB implementations are available for different platforms from several vendors, including some license-free products. CORBA 2.3 or higher is required; it guarantees the interoperability of ORBs from different vendors.

For more information on CORBA and the OMG please refer to the appropriate documentation and literature available (www.omg.org, www.corba.org).

The specification of the OO-API uses CORBA IDL (Interface Definition Language). There are language mappings to several common programming languages (like C, C++, JAVA) and IDL compilers that translate the interface specifications into e.g. C++ header files or JAVA files which can immediately be used to produce client stub or server skeleton code.

Upward compatibility is guaranteed in that existing interface definitions will only be extended if changed in the future.

The following sections describe the main aspects of the OO-API. This is not more than a brief overview, does not cover all interfaces, and it is strongly recommended to read the detailed specification documents to further learn about how to apply the API.

THE ENTRY POINT

As is state of the art for modern systems architectures, the API provides a factory interface

```
AoFactory {...}
```

Whenever an ASAM-ODS server is available in the network and registered with an ORB, it will expose this factory interface and allow a client to ask for general information on the server as well as to initiate a session with that server (by using an authentication string with user name and password).

A session itself exposes an interface

```
AoSession {...}
```

The main purpose of a session is to

- provide access to the base model, application model, and instances within the ASAM-ODS server (this includes a reference to those interfaces that allow to define and execute queries that further narrow the interesting set of instances),
- provide access to possible options (so-called context variables) regarding the behavior of the ASAM-ODS server, and some default settings when working with instance elements.
- manage transactions and thus allow the client to invoke a sequence of interface actions that must either succeed or must be withdrawn by the server as a whole.

The session interface allows to access the application model via the interface ApplicationStructure (described below) either by returning a reference to it or by completely returning it by value. This is finally an issue of server and network load, and it is up to the client to decide whether during this session only a part of the application structure or the whole information will likely be needed.



ACCESS TO THE BASE MODEL

Access to the base model is provided through the interfaces

BaseStructure {...}
BaseElement {...}
BaseAttribute {...}
BaseRelation {...}

A reference to the base model can be requested from the session.

The BaseStructure interface allows to access all base elements and all base relations (between two base elements); the client may specify several selection criteria ('patterns') to browse through the base model as is needed.

The version number of the ASAM-ODS base model is available from this interface as well.

A base element knows about (and can be requested to provide) all its base attributes, its base relations, and those other base elements to whom it is related.

A base attribute knows about (and can be requested to provide) its data type, name, whether it is a mandatory attribute, and others. It can only be accessed through the base element where it belongs to.

A base relation can be viewed as a special kind of base attribute with a specific interface. It describes a directed link between two base elements and holds (besides others) a reference to the first and second base element (start- and endpoint), the type of the relation (e.g. FATHER, CHILD, INFO, ...), and the range of the relation (e.g. [0,1], [1,MANY], ...). It knows about an inverse relation (where the elements are the same but the direction is opposite), if one exists.

These interfaces of the base model provide all information on the specific objects, but do not allow to modify them.

ACCESS TO THE APPLICATION MODEL

Access to the application model is provided through the interfaces

ApplicationStructure {...}
ApplicationElement {...}
ApplicationAttribute {...}
ApplicationRelation {...}

A reference to the application model can be requested from the session.

The ApplicationStructure interface allows to access all application elements and all application relations (between two application elements); the client may specify several selection criteria ('patterns') to browse through the application model as is needed.



Additionally the ApplicationStructure interface allows to add or remove application elements and application relations.

The ApplicationStructure interface provides access to instances, again allowing to specify selection criteria that shall be applied by the ASAM ODS server.

Finally, the client may ask the application model to check itself for consistency.

An application element allows a variety of actions, they include:

- browse through the application structure, finding related application elements, application relations, instances, corresponding base element, ...
- create or delete application attributes or instances of this application element
- request or specify the access rights to the application element and its instances

An application attribute knows about (and can be requested to provide) its data type, name, whether it is a mandatory attribute, and others. It will be linked to a specific unit; access rights may be specified, allowing only restricted access to the application attribute. In contrast to a base attribute, all this information may be set initially or changed later by a client.

An application relation can (similarly to a base relation) be viewed as a special kind of application attribute with a specific interface, holding the same kind of information as a base relation above. It specifies a directed link between two application elements. However, a client may not only request information on the application relation but may change it.

ACCESS TO GENERAL INSTANCES

The application elements and the application model itself can be used to get a reference to instances. In case more than one instance is referenced, an iterator interface is provided to ease the navigation through the list of instances.

The instances themselves provide the interface

```
InstanceElement {...}
```

In case a set of instances has to be worked with, this can be done through the interface

```
InstanceElementIterator {...}
```

Instances may be asked for their name, ID, relations to other instances and other characteristics.

Relations may be set to other instances. Additional attributes may be added to or removed from an instance, and the instance attributes may be set to specific values (with units).

Furthermore there are methods to handle the access rights to the instance.

ACCESS TO RESULT DATA

Result data are typically mass data and are therefore treated differently. They are kept within the ASAM-ODS storage as instances, but these instances show a specific behavior and therefore are accessed through the following special interfaces:



Measurement {...}
SubMatrix {...}
ValueMatrix {...}
SMatLink {...}
Column {...}

The interfaces 'Measurement' and 'SubMatrix' show the behavior of instances (are derived from the instance interface) but allow additional actions to be performed.

In order to use these five interfaces, it is important to understand how result data are organized:

The results of a measurement are values of quantities at subsequent measurement points. However not all quantities are necessarily sampled at the same measurement points; some may be sampled at a higher rate/density than others; some may be sampled only during a part of the whole duration of the measurement. Storing the measurement in one (large) matrix may leave large parts of that matrix empty, which results in a waste of memory.

Therefore ASAM-ODS allows to store results of one measurement into a set of matrices (called 'submatrices'). Each of them contains data of quantities that are sampled in the same way and thus don't lead to irrelevant (empty) matrix positions.

The access to those result data can be manifold:

- The complete measurement data can be accessed through the 'Measurement'-interface. In this case a ValueMatrix is returned, which then can be asked to provide one or more result data (including its units). It is possible to either request all or part of the data of the same measurement point (row) or all or part of the data of the same measurement quantity (column). Data may be added, modified, or removed. This includes adding or removing complete rows and columns of the matrix.
- Data of one submatrix may be accessed through the 'SubMatrix'-interface. Similarly to above, a ValueMatrix is returned which then can be asked to provide one or more result data (including its units). However, this value matrix will only provide data from the submatrix.
- How the individual submatrices have to be combined to get a (partly empty) complete measurement matrix, can be set as well as requested by using the 'SMatLink'-interface.
- The meaning of a column within a submatrix as well as additional information on the column can be found using the 'Column'-interface.

QUERIES

The ASAM-ODS API allows a client application to narrow the information it requests from a server through a set of conditions. Only those instances are returned by the server that comply to the conditions. This method to work with huge amounts of data is called 'query'.

Conditions are specified through the use of comparison operators like '>', '<', '=' (including operators like 'contained in set' or 'not contained in set' for e.g. enumeration type attributes)



together with logical operators like AND, OR, and NOT (with the possibility to place brackets).

Queries are performed by a server in synchronous mode; the client must wait until the server has completed the query execution.

In order to process queries the following interface is exposed by an ASAM-ODS server:

ApplElemAccess {...}

1.6.2 THE PROCEDURAL API (RPC-API)

Compared to the OO-API, the RPC-API has only a few methods that may be invoked by a client and will be submitted to the server through a network using the ONC-RPC (open network computing remote procedure call) mechanisms. Quite sophisticated structures are typically handed over when calling the methods; the server itself returns the service results also by means of structures.

The principal sequence a client must obey to contact and use a server is similar to the OO-API:

- The client first has to create a session using AOP_OpenEnv(..)
- The client may then request the application model using AOP_GetApplInf(..). This returns the complete application model including a list of all application elements and relations. He may ask for information on all application attributes of each of the application elements through AOP_GetAttr(..).
- The client may request a list of all instances of an application element by calling AOP_GetInstRef(..).
- The client may request the values of application or instance attributes with AOP_GetVal(..), AOP_GetInstAttr(..), and AOP_GetValE(..). Selection mechanisms are built in so that only those values are transferred that match user-specific criteria.
- The client may delete or add instances or modify attribute values using AOP_PutVal(..).
- Measurement data are read from or fed into the server using specific methods that work on submatrices. To access them is done through AOP_GetValAttr(..), AOP_GetValInf(..), AOP_GetValVal(..), AOP_PutValVal(..).
- Finally the client must close the session with AOP_CloseEnv(..).

There are some more methods that allow to set or retrieve security information (and thus restrict the access to specific user groups), to manage personal settings, and for other purposes.

Though there are minor extensions in the interface definition file compared to earlier versions, there has been no change in the methods and their parameters since version 3.2 of the RPC-API, and no changes are planned in the future.

Interoperability is guaranteed with only few restrictions as



INTRODUCTION To ASAM ODS

- old clients will still work with a server that builds upon the latest interface definition though they may not be able to access the latest base elements, since the corresponding base identifiers will probably not be known to them.
- new clients will work with both new and old servers; however it may happen that they cannot access the latest base elements when connecting to old servers, since the base identifiers will probably not be known to the old servers.

Both restrictions might be fixed by a minor change to any old client or server.

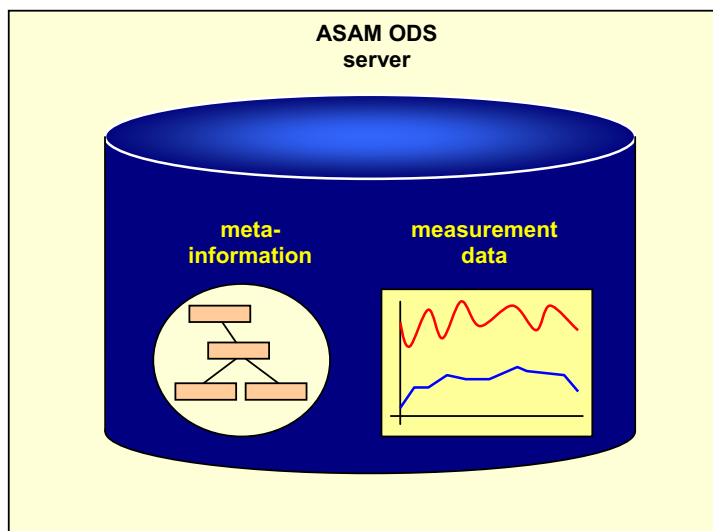


1.7 THE PHYSICAL DATA STORAGE - A DEEPER INSIGHT

This section will give more detailed information on the third aspect of the ASAM ODS standard. However, to implement this standard will require to study chapter 3 of the ASAM ODS Version 5.0 documentation.

Currently there is a specification for a physical storage format using relational databases and mixed-mode storage. While all data are stored within the database tables according to the basic ASAM ODS physical data storage, the mixed mode storage allows to source out mass data into separate files.

Storing information using a relational database requires to specify which tables must be available in the database, what columns they need to have, which table entries must be unique (because they are used as keys) etc.. This has been done for ASAM ODS information and is roughly described in this section.



First of all, as the figure indicates, a distinction must be made between the storage of meta-information and the storage of real values (that is: all measurement data as well as the instance elements' attribute values and relations).

Information on the base model is not stored at all in the database; instead it is kept somewhere (e.g. in the source code) within the ODS server.

Information on the application model is stored using mainly three tables:

a) the application elements table (also referenced as **SVCENT**)

This table contains a list of all application elements and for each of them allows to keep the information on

- the identifier of the application element (which is provided by the ODS server and must be unique within this table),
- its name (given by the user/application model designer),
- the identifier (BE-ID) of the base element to which this application element belongs (which is uniquely defined by ASAM ODS when specifying the base model),
- the name of the table where the instances of that application element are stored.



INTRODUCTION TO ASAM ODS

The figure shows the structure of this table, together with some example entries:

AE-ID	AE-Name	BE-ID	InstTabName
1	Acquisition	3	AcqInst
2	Vehicle	21	VehicleInst
3	Engine	22	EngInst
4	Cylinder	22	CylInst
...	...	...	...

b) the application attributes table (also referenced as SVCATTR)

This table contains a list of the application attributes of all application elements. It also includes all application relations that are [0,n]- or [1,n]-relations. Its structure allows to store information like

- the name of the application attribute or relation (AA/AR-Name) (which is supplied by the application model designer and must be unique within the application attributes/relations of the same application element),
- the identifier (AE-ID) of the corresponding application element; this is the same as given in table SVCENT,
- (in case this application attribute shall represent a base attribute:) the name of the corresponding base attribute (BA-Name) (this name is defined by ASAM ODS when specifying the base model),
- (in case the entry describes an application relation:) the identifier of the application element to which this application relation points (AET-ID, target),
- the name of the column of the instances table, where the values of this application attribute/relation are contained,
- the identifier (DT-ID) of the data type with which this application attribute/relation is stored in the instances table,

The figure shows the structure of this table, together with some example entries:

AE-ID	AA/AR-Name	BA-Name	AET-ID	InstColName	DT-ID	...
1	AcqName	name		AcqNCol	1	...
1	AcqID	id		AcqIDCol	6	...
1	MeaBegin	measurement_begin		MeaBCol	10	...
1	MeaEnd	measurement_end		MeaECol	10	...
1	Acq_Eng	units_under_test	3	Acq_Eng_Col	6	...
3	Name	name		NameCol	1	...
3	ID	id		IDCol	6	...
3	NumOfCyl			NumOfCCol	2	...
3	Eng_Acq	measurement	1	Eng_Acq_Col	6	...
3	Eng_Cyl	children	4	Eng_Cyl_Col	6	...
4	Name	name		NameCol	1	...
4	ID	id		IDCol	6	...
4	Bore			BoreCol	3	...
4	Stroke			StrokCol	3	...
4	IgnitionDelay			IgnDelCol	3	...
4	Cyl_Eng	parent_unit_under_test	3	Cyl_Eng_Col	6	...
...	...	...	...	...	...	...



c) the application relations table (also referenced as SVCREF)

This table contains all relations between application elements that are of type $[n,m]$. Relations of type $[0,n]$ or $[1,n]$ can be handled like attributes and are contained in the above described table SVCATTR. The table SVCREF contains information on

- the identifier (AE1-ID) of one of the two application elements that have a $[n,m]$ -relation
- the identifier (AE2-ID) of the other application element within this $[n,m]$ -relation,
- the name of the table where each single relation instance is stored.

Separated from these three (more or less static) tables are a set of tables that contain the real values (the instances):

A) Tables containing instance elements and their attributes

There is one table for each application element, holding its instances. The name of the table is given in SVCENT.

Each column contains the values for one application attribute/relation. The relationship between the application attribute/relation and the corresponding column is given by the column name and specified in SVCATTR.

Each row contains one instance of that application element.

The figure shows a simple example for such a table ("CylInst"; the IDs are usually server-defined):

NameCol	IDCol	BoreCol	StrokCol	IgnDelCol	Cyl_Eng_Col
Cyl1	100	80.0	75.0	0	110
Cyl2	101	80.0	75.0	540	110
Cyl3	102	80.0	75.0	180	110
Cyl4	103	80.0	75.0	360	110

B) Tables containing instance relations in case of $[n,m]$ -relation types

There is one table for each $[n,m]$ -type application relation within the application model.

Each of the two columns contains instance elements of only one application element.

Each row specifies which two instance elements are related. The same instance element will typically appear more than once in the table.

C) Tables containing instance attributes that don't have corresponding application attributes

Attributes/relations that are added to instances and thus do not exist in the corresponding application elements cannot be contained in the tables according to A). Therefore a separate table (named SVCINST) is set up to hold all these instance attributes/relations. Each row contains one instance attribute/relation; the columns store information about the name, the unit identifier, the value, and the identifier of the instance element they belong to.

**D) Tables containing measurement results (mass data)**

There are tables in the database that hold the measurement data. Measurement data are composed of several submatrices that together build up an (usually sparse) overall measurement matrix. Each column of a submatrix (a so-called "local column") contains data of one measurement quantity. Each row of a submatrix contains data of all measurement quantities sampled at the same measurement point (typically point of time).

E) Tables containing security information

To restrict or allow access to ASAM ODS data, the physical storage holds three tables: SVCACLI, SVCACLA and SVCTPLI. SVCACLA holds the security data for the instance protection, SVCACLA holds the security data for the attribute and element protection, and SVCTPLI holds the ACL templates for security.



1.8 THE ASAM TRANSPORT FORMAT (ATF) - A DEEPER INSIGHT

This section will give more detailed information on the fourth aspect of the ASAM ODS standard. However, to implement this standard will require to study chapter 5 (resp. 6) of the ASAM ODS Version 5.0 documentation.

The ATF is a specification of a syntax to describe ASAM ODS information so that it can be transported as a text file. It is typically used to

- transport whole or part of the contents of one ODS database to another one.
- import information from a system into an ODS database in case that system is not capable to interface to the server through the ODS API or is currently not connected to the server through a network.

ATF allows to transport the complete application model as well as the instance elements with their attributes and relations. Since the base model is standardized it is not necessary to include it in the ATF. In order to handle changes in the base model standard over time, provisions will be made to include the assumed base model version in ATF.

ATF is specified as a plain ASCII text format (characters 32₁₀..255₁₀). Thus it is very likely that it can be transported onto every available platform (operating system). Advanced versions of the ATF specification will probably base on the UTF8 character set.

The ATF specification allows to split the whole information into a set of individual files, each one except the first being 'included' by another file. This has several benefits:

- If the transport media are limited regarding file size, transport can take place in several consecutive steps.
- If the transport takes place over a noisy channel, it is often easier to transfer several small files than one rather large file.
- If parts of the whole information are always constant (e.g. the application model, or some descriptive data) only the variable parts of the information require additional disk space; the constant parts can be included wherever they are needed.

The ATF specification allows to source out mass data, which typically appear when LocalColumn information (the real measured values) have to be transported. Since transporting data in ASCII format requires several times the space than binary data, ATF allows to transport mass data in binary format. For this purpose the corresponding ATF ASCII file contains a specification of the structure of those binary files as well as information on the sequence of bytes for standard data types (i.e. is high or low byte written first to the file stream? etc.).

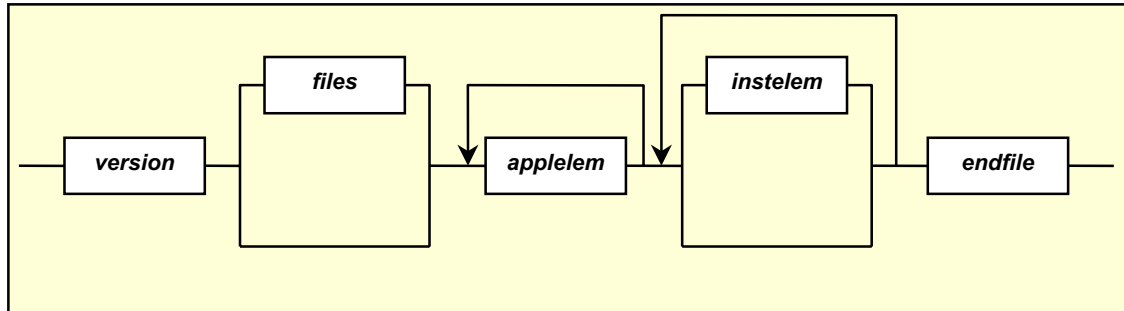
There are two different ATF specifications contained within the ASAM ODS Version 5.0 documentation: The classic ATF/CLA and the new ATF/XML. While the ATF/CLA is a proven standard for data exchange and has been used for years, the ATF/XML is expected to benefit from the expanding XML technology.

The ATF/CLA will still be supported by ASAM ODS in the future, though no further revisions of that ATF type are expected to come up.



1.8.1 ATF/CLA

The syntax of the classic ATF is completely specified in chapter 5 using syntax diagrams. The macroscopic structure of ATF/CLA is shown in the following figure.



A (logical) ATF/CLA file is divided into five main parts, as shown:

- version: provides information on the ATF/CLA version.
- files (optional): provides a list of physical files that together build the complete logical ATF/CLA file.
- applelem: is a description of an application element and may be repeated for each application element that is going to be transported by this ATF/CLA file.
- instelem (optional): is a description of an instance of an application element; it appears once for each instance that is transported by this ATF/CLA file. In case the ATF/CLA file carries only meta-information, the number of instelem is zero.
- endfile: provides the information that there is no further ATF-relevant information in the file.

Each of these syntax elements is further specified in the ATF/CLA documentation down to a level of sequences of single characters out of the ASCII character set.

Attention must be paid to semantics. The ATF/CLA specification is a syntax specification and does not care much about semantics. However any application working with ATF/CLA files does care.

E.g. according to the ATF/CLA syntax definition it is possible to create a syntactically correct ATF/CLA file

- with an instance element that is related to a non-existing application element, or
- with an application element that does not contain mandatory base attributes, or
- with an application or instance element that has no relation to any other element and is not related by any other element in the ATF/CLA file.

All these examples are conform to the syntax, but importing them into an ODS server will immediately create problems.

Thus there is no way to generate a proper ATF/CLA file without knowing about the ASAM ODS data model.



1.8.2 ATF/XML

Since with XML standards for text-based data exchange have come up during the past few years, the ASAM ODS workgroup also defined a standard for ATF based on XML. This exchange format is completely specified in chapter 6 using XML schema.

For an easy upgrade the keywords of ATF/XML are defined close to those for ATF/CLA.

XML tools allow for a more detailed automatic verification of an ATF/XML file than is possible with ATF/CLA.



1.9 APPLICATION MODELS - A DEEPER INSIGHT

This section will give more detailed information on the fifth aspect of the ASAM ODS standard. However, to implement this standard will require to study chapters 11 to 13 of the ASAM ODS Version 5.0 documentation.

Whenever different products base on the same application model data may be exchanged between them very easily. A common application model guarantees that products store the same information with the same data formats in the same way, and thus the meaning of this information can be generally understood.

Currently three specific application models have been specified and released by ASAM ODS. They cover areas that are of great importance in the automotive industry. These models are based on the ASAM ODS base model, and are intended to be minimal application models. This means that anyone who creates a company specific application model can use the proposed model and add company specific items.

1.9.1 THE NVH APPLICATION MODEL

NVH stands for 'Noise, Vibration and Harshness', and it is a specific field of activity in the industry, where a lot of simulation results and measurement data exist. Since there exist quite a lot of software solutions for problem solving in the NVH field, being able to share data between all these solutions is of high importance to the users of such software.

The data that are typically measured and processed in this domain are very diverse in nature, and in order to be able to correctly interpret these data, some descriptive information is typically added to the data in form of extra parameters.

1.9.2 THE CALIBRATION APPLICATION MODEL

The calibration data model is intended as a schema to structure calibration data obtained from the calibration process of test stand components like sensors, amplifiers, ... etc. The main usage of an ASAM ODS server is to store and retrieve measuring data of different kind (vehicle data, engine data, NVH data, ... etc.). A measuring data analysis relies implicitly on a perfectly calibrated test bed, respectively its measuring components operating permanently fault free in same quality over arbitrary long periods. This application model defines how calibration information should be stored in an ASAM ODS server.

1.9.3 THE VSIM APPLICATION MODEL

VSIM stands for 'Vehicle Safety Information Model'. The VSIM application model standardizes the data storage for the area of vehicle safety tests. It specifies a list of application elements and their attributes and relations. It includes mechanisms to describe and reference externally stored diagrams, test reports, photos, and movies of the tests and relate them to the test data within the ASAM ODS server.

The VSIM application model has already been specified in 2001 within the ISO TC22/SC12/WG3 as ISO/TS 13499. It is thus not contained in the current ASAM ODS documentation and may be requested from the ISO.



1.10 STATUS AND FUTURE STEPS

1.10.1 STATUS

ASAM ODS has released stable specifications for the ASAM ODS data model (using STEP EXPRESS), the physical storage for relational databases and mixed-mode storages, the interfaces (using RPC and CORBA IDL), and the transport format.

Additionally, ASAM ODS has prepared application-specific data models, e.g. for NVH (Noise, Vibration and Harshness), calibration data, VSIM, etc.. More models are expected to come up in the future.

For detail information contact the ASAM office (e.g. via the ASAM website www.asam.net).

Several implementations (ASAM ODS Servers, data browsers, evaluation packages, and other tools) based on these specifications are available from different vendors and are used in projects in the automotive industry. Please contact the ASAM office or look into the ASAM website to find companies involved in the ASAM standardization process.

Over the time, ASAM ODS has proven to be a very stable standard. Based on the experience made in the last years, there is no need for excessive changes or extensions of the kernel structure. ASAM ODS is now concentrating on applications and data models for applications (see above).

1.10.2 FUTURE STEPS

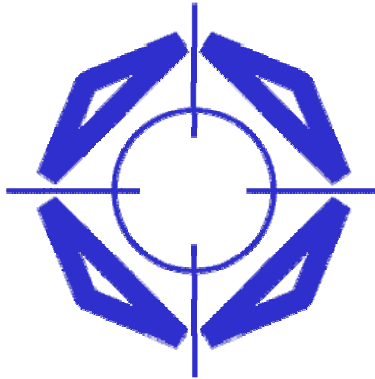
ASAM ODS will continuously keep on improving and consolidating its standards and will continue to observe the IT trends and integrate applicable upcoming standards.

At the time of publishing this document, ASAM ODS is undergoing the so-called Harvesting Procedure within the ISO group TC184/SC4 to become an ISO standard in the future.



1.11 REVISION HISTORY

Date Editor	Changes
2003-10-10 R. Bartz	ATF/XML has been introduced. The APIs have been explained in more detail. The RPC Interface has been included. The description of the Query capabilities has been changed. Specific application models are introduced: Calibration, NVH. Minor textual changes have been made.
2003-10-17	The FTR meeting agreed to the current text with two modifications required
2003-10-18 R. Bartz	A figure to further explain the base model has been included.
2003-12 R. Bartz	A reference to the VSIM model has been made. Minor textual changes have been introduced.
2003-12-30 R. Bartz	The Release version has been created
2004-09 R. Bartz	Minor textual changes have been introduced



ASAM e.V.
Arnikastr. 2
D-85635 Hoehenkirchen
Germany

phone: (+49) 8102 – 895317
fax: (+49) 8102 – 895310
e-mail: info@asam.net
internet: www.asam.net

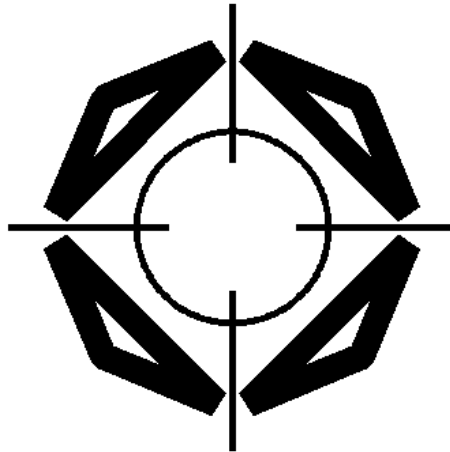
ASAM ODS

VERSION 5.0

ISO-PAS

CHAPTER 2

ARCHITECTURE



**Association for Standardization of
Automation and Measuring Systems**

Dated: 30.09.2004

© ASAM e.V.

Status of Document

Reference:	ASAM ODS Version 5.0 Architecture
Date:	30.09.2004
Author:	Peter Dornhofer, AVL; Horst Fiedler, TIFFF; Dr. Helmut Helpenstein, National Instruments; Gerald Sammer, AVL; Karst Schaap, HighQSoft
Type:	Specification
Doc-ID:	ASAM_ODS_50_CH02_Architecture.PDF
Revision Status:	Release

Note: ASAM ODS has invoked a Formal Technical Review (FTR) process which intends to continuously improve the quality and timeliness of its specifications. Whenever an error is identified or a question arises from this specification, a corresponding note should be sent to ASAM (odsfr@asam.net) to make sure this issue will be addressed within the next review cycle.

Disclaimer of Warranty

Although this document was created with the utmost care it cannot be guaranteed that it is completely free of errors or inconsistencies.

ASAM e.V. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any expressed or implied warranties of merchantability or fitness for any particular purpose. Neither ASAM nor the author(s) therefore accept any liability for damages or other consequences that arise from the use of this document.

ASAM e.V. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.