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)

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order to decide whether it will be confirmed for a further three years, revised to become an International Standard, or withdrawn.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again after a further three years, at which time it must either be transformed into an International Standard or be withdrawn.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 **S**tandardization of
Automation and **M**easuring Systems

Dated: 30.09.2004

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 (odsftr@asam.net) to make sure this issue will be addressed within the next review cycle.

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.

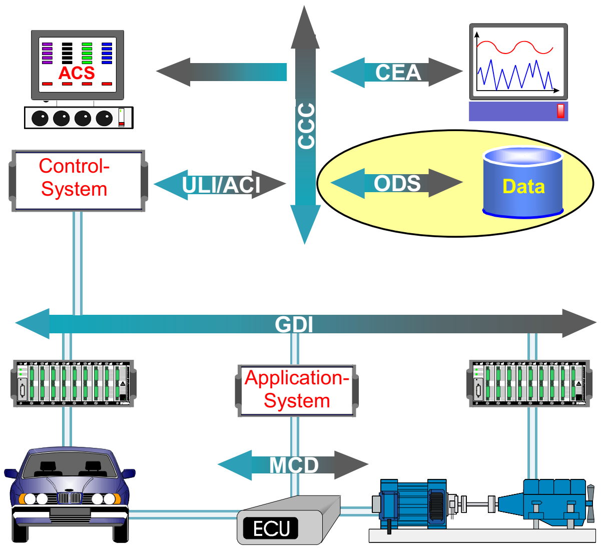
This chapter must **NOT** be seen as a specification; it is an introduction with some details to provide an overview to the interested reader.

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.

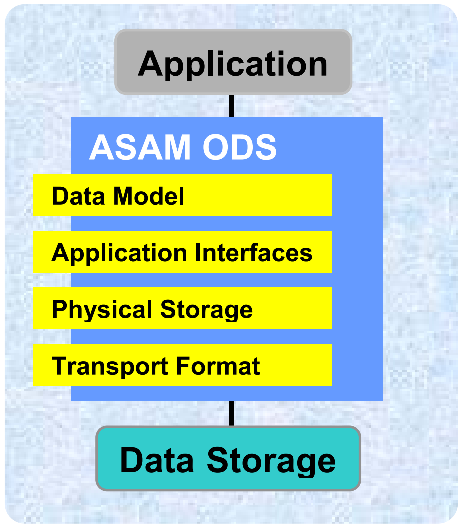
Moreover different customers require a different interaction scenario. Product releases become customer specific, version control becomes a major job.

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.

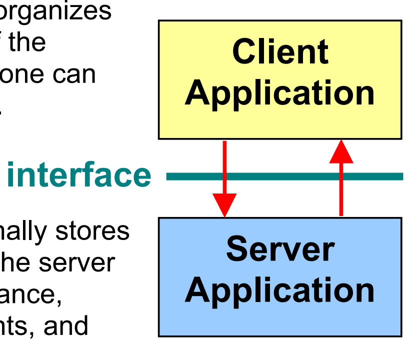
Data in relational databases are organized in tables which may be related to each other. Thus specification work has to concentrate on specifying the tables and their relationships. ASAM ODS has done so (see the 'physical storage' description below).

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 loosing 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 data-bases. 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)

- 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.

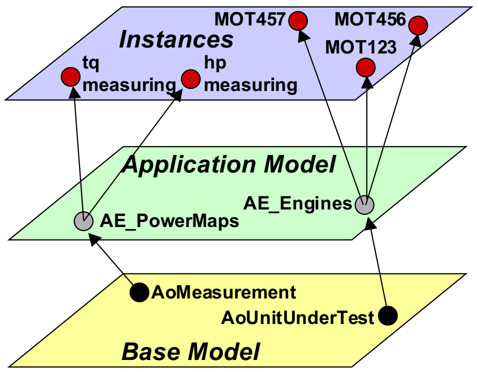
1.4 TECHNOLOGICAL LEVEL FOR IMPLEMENTATION

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
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

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.

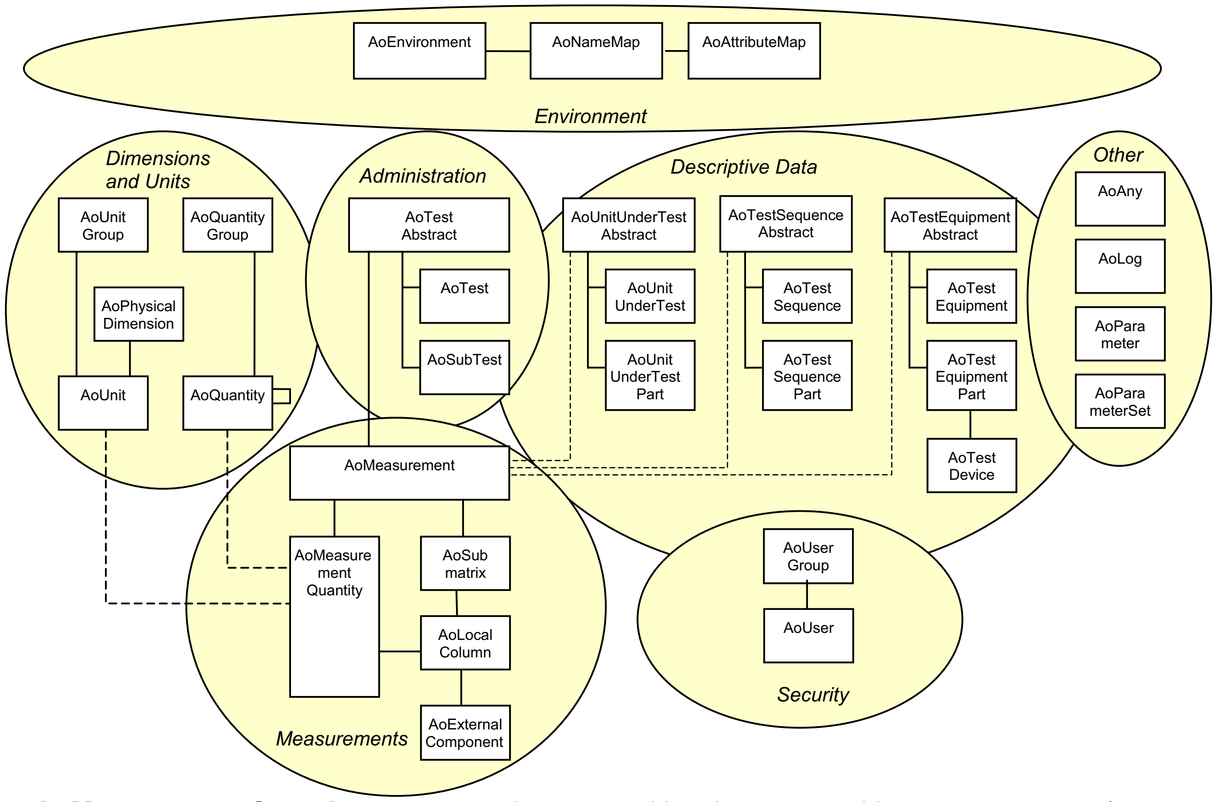
Base relations are links between those base elements; each of these links carries a specific meaning.

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.



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.

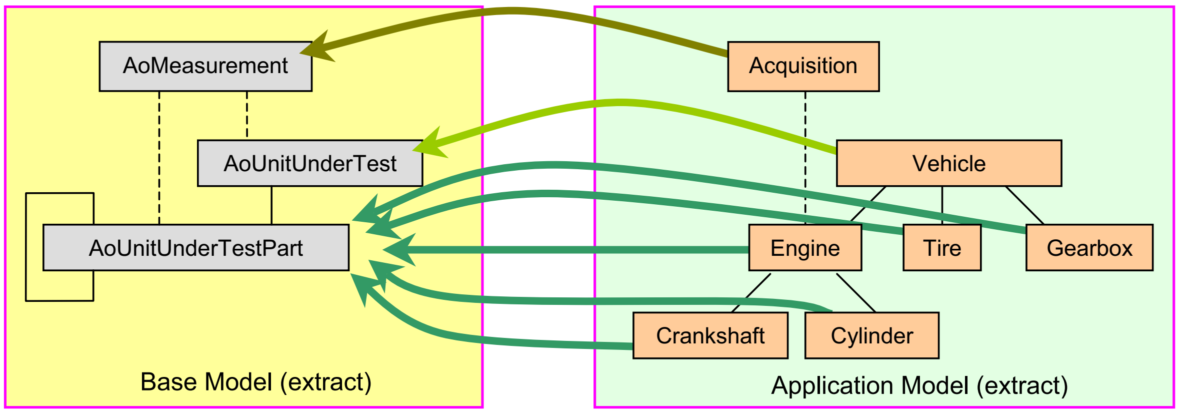
Each base element has at least the base attributes 'name' and 'ID' and may have attributes like 'version', 'version_date' and 'description'.

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.

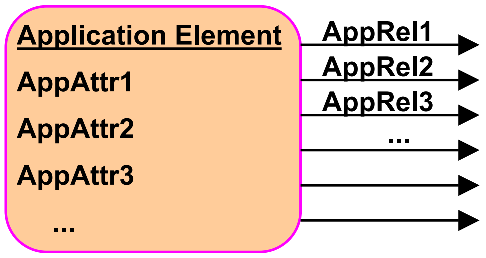
The same is true for application attributes: Each application element has a set of attributes that quantify its characteristics. The mandatory base attributes must also appear in each corresponding application element, the optional base attributes may appear in the application elements.

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

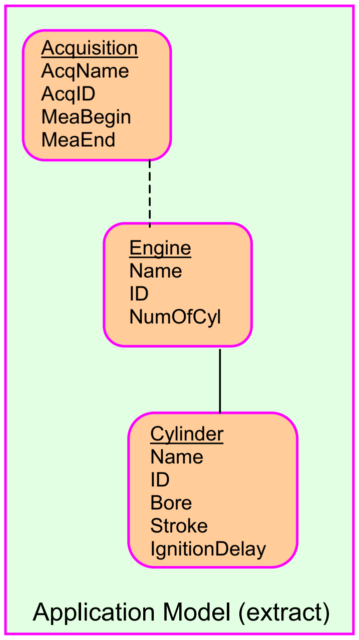
Once an application model is designed, the structure of the data stored/retrieved is defined. Neither base model nor application model store any (measured or descriptive) values. They just provide '<u>meta-information</u>' to know about the structure of the data store.

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):

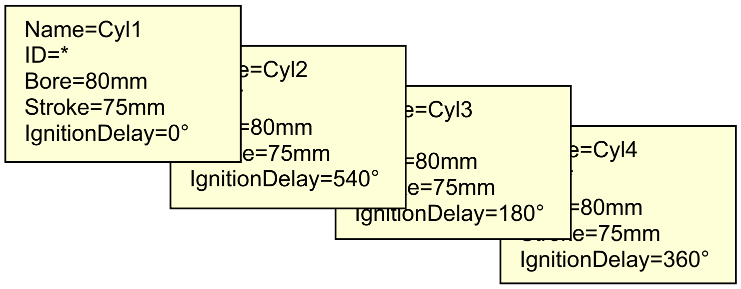


➢ one instance element of application element 'Engine':

Name=NewEngine

ID=*

➢ four instance elements of application element 'Cylinder' :



➢ one instance element of application element 'Acquisition'

AcqName=Acq1
AcqID=*
MeaBegin=22.Jun.00 23:00
MeaEnd=22.Jun.00 23:07

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*.

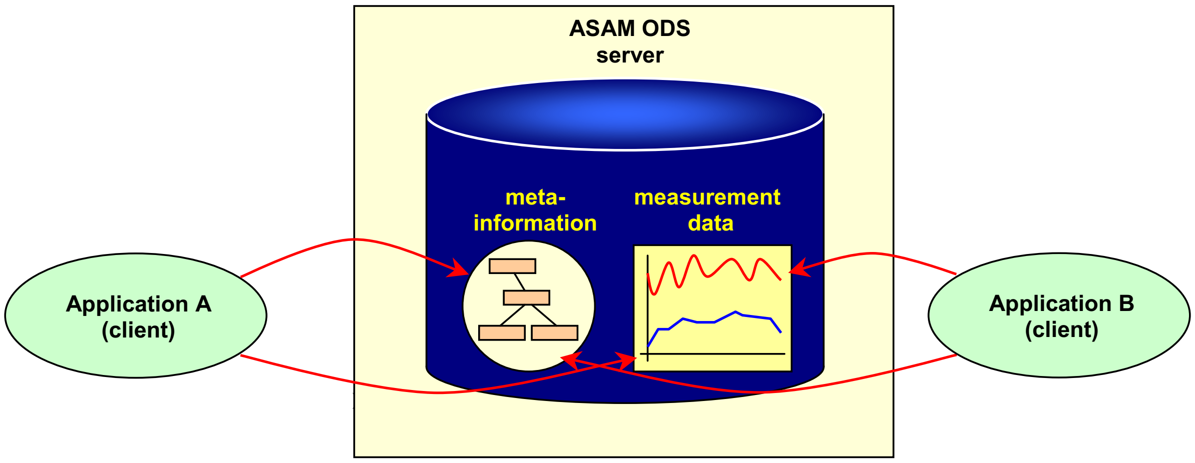
All modules and functions of the OO-API are explained in detail in chapter 10 (including purpose, calling sequence, return values, and example code) . All methods defined for the RPC-API are described in chapter 9.

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 Request 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

- 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.

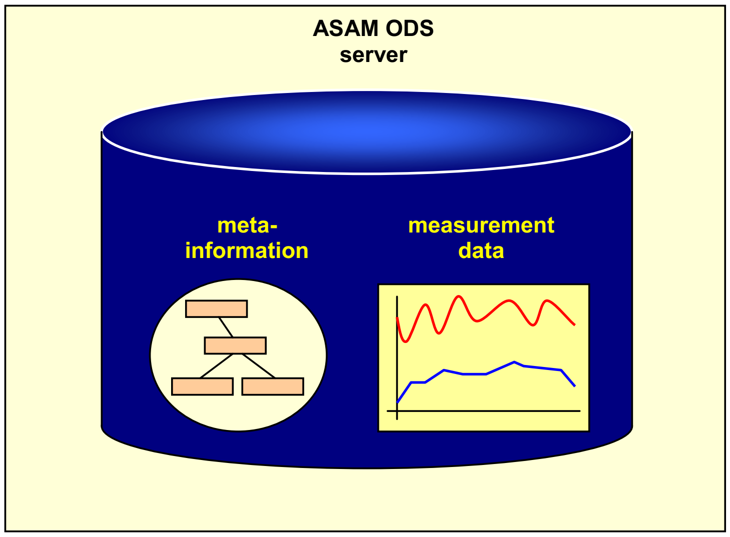
Both restrictions might be fixed by a minor change to any old client or server.

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.

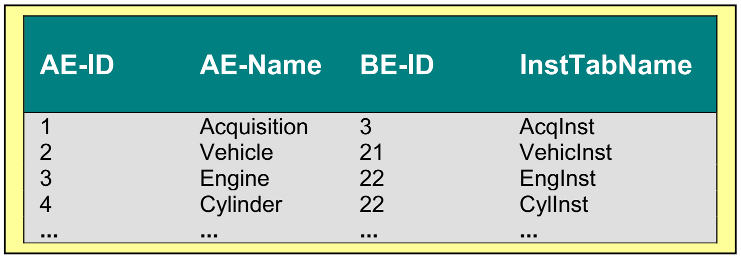
Information on the application model is stored using mainly three tables:

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.

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



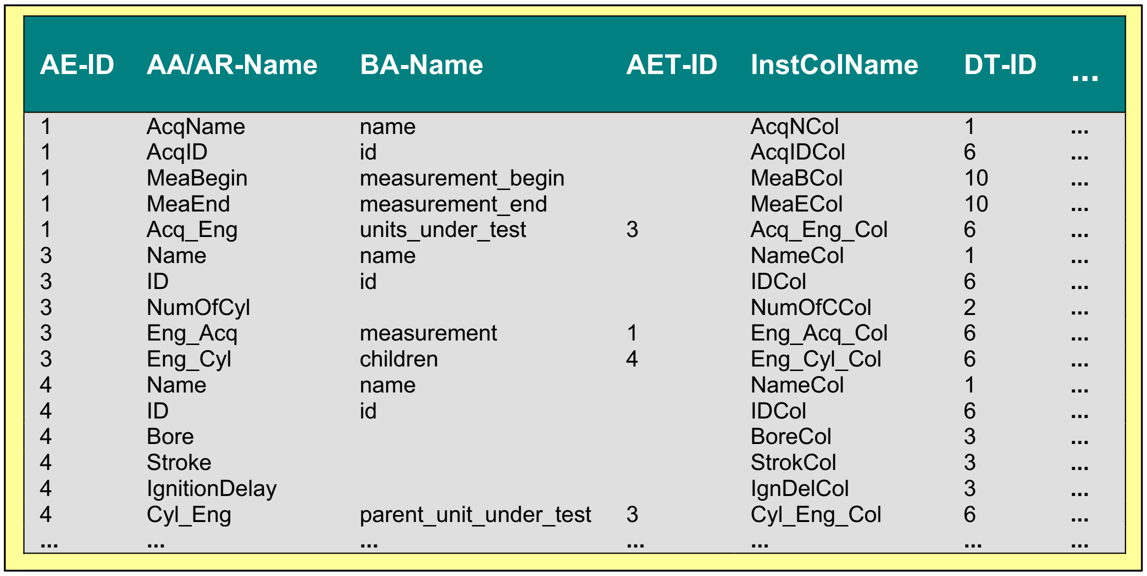
AE-ID	AE-Name	BE-ID	InstTabName
1	Acquisition	3	AcqInst
2	Vehicle	21	VehicInst
3	Engine	22	EngInst
4	Cylinder	22	CylInst
...	...	...	...

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):

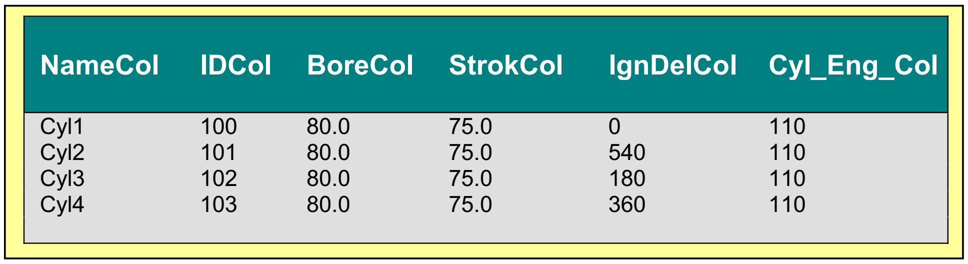
A) Tables containing instance elements and their attributes

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 $32_{10}..255_{10}$). 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.

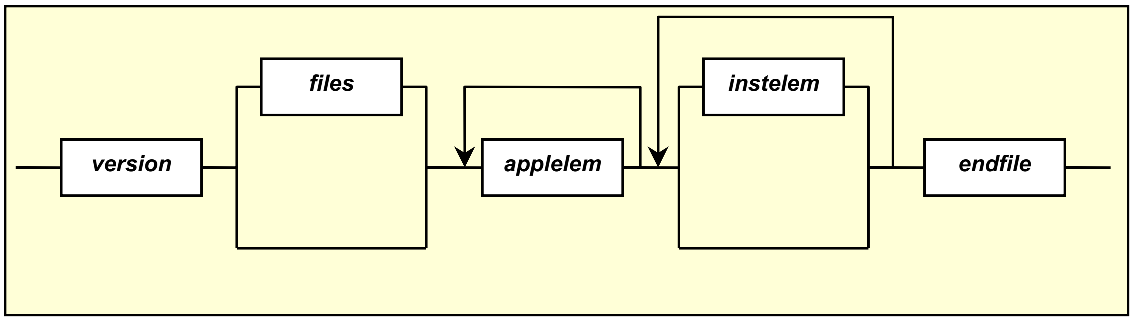
The ATF specification allows to source out mass data, which typically appear when LocalColumn information (the real measured values) have to be transported. Since transporting data in ASCII format requires several times the space than binary data, ATF allows to transport mass data in binary format. For 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 **S**tandardization of
Automation and **M**easuring Systems

Dated: 30.09.2004

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 (odsftr@asam.net) to make sure this issue will be addressed within the next review cycle.

Contents

2 ASAM ODS ARCHITECTURE 2-5

2.1 INTRODUCTION 2-5
2.2 ELEMENTS OF THE BASE MODEL 2-7
2.2.1 OVERVIEW 2-7
2.2.2 ENVIRONMENT 2-7
2.2.3 DIMENSIONS AND UNITS 2-8
2.2.4 ADMINISTRATION 2-8
2.2.5 MEASUREMENTS 2-8
2.2.6 DESCRIPTIVE DATA 2-8
2.2.7 SECURITY 2-9
2.2.8 OTHER DATA 2-9
2.3 RULES FOR THE APPLICATION MODEL 2-10
2.3.1 NAMES OF APPLICATION ELEMENTS 2-10
2.3.2 BUILDING THE APPLICATION MODEL 2-10
2.3.3 MAPPING BASE MODEL AND APPLICATION MODEL 2-13
2.4 ATTRIBUTES IN ODS MODELS 2-14
2.4.1 RULES FOR ATTRIBUTES 2-14
2.4.2 ATTRIBUTES OF ALL ELEMENTS 2-14
2.5 DATA TYPE USAGE IN ASAM ODS 2-15
2.5.1 GENERAL NOTES TO DATA TYPES IN ASAM ODS 2-15
2.5.2 DATA TYPES OF THE BASE MODEL AND THE APIs 2-17
2.5.3 ALTERNATIVES TO DATA TYPES OF BASE ATTRIBUTES 2-18
2.5.4 ALTERNATIVES TO RELATION RANGES OF BASE RELATIONS 2-19
2.5.5 THE STANDARDIZED ASAM DATA TYPES 2-19
2.6 THE APPLICATION INTERFACES 2-22
2.6.1 BASIC INFORMATION 2-22
2.6.2 WORKING WITH THE APPLICATION INTERFACE 2-27
2.7 INHERITANCE SUPPORT 2-33
2.7.1 SOFTWARE DESIGN CLARIFICATIONS AND LIMITATIONS 2-33
2.7.2 EFFECTS WHEN USING INHERITANCE 2-33
2.7.3 APPLICATION MODEL 2-34
2.7.4 PHYSICAL STORAGE 2-34
2.7.5 RESTRICTIONS OF THE RPC-API 2-36
2.7.6 INSTANCE ATTRIBUTES 2-37
2.7.7 RELATIONS 2-37
2.8 THE USE OF ATF IN THE ODS ARCHITECTURE 2-38
2.9 SECURITY CONCEPTS OF ASAM ODS 2-39
2.9.1 GENERAL 2-39

2.9.2 BASIC CONCEPT 2-39
2.9.3 EXPLANATION OF BASIC TERMS 2-39
2.9.4 ACCESS CONTROL FOR DATA OBJECTS 2-40
2.9.5 BASIC RIGHTS 2-44
2.9.6 CHECKING THE ACCESS RIGHTS FOR A DATA OBJECT 2-46
2.9.7 CREATING A NEW DATA OBJECT 2-46
2.9.8 OTHER BASIC RULES 2-48
2.9.9 PHYSICAL DATA MODEL 2-50
2.9.10 ACL 2-50
2.9.11 CONFIGURATION OF ACL PROTECTION 2-51
2.9.12 ACL TEMPLATE 2-51
2.9.13 USERS AND USER GROUPS 2-52
2.9.14 ATF FORMAT 2-52
2.10 GLOSSARY 2-54
2.11 REVISION HISTORY 2-56

Scope

This document describes the Architecture of ASAM ODS Version 5.0.

Intended Audience

This document is intended for implementers of ASAM ODS Version 5.0. It shall be used as a reference on the architecture of ASAM ODS. It describes in detail the base elements with their corresponding base attributes, base relations etc..

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
- ANSI SQL89: Database Language - SQL with Integrity Enhancements. American National Standard X3.135, American National Standards Institute, 1989.

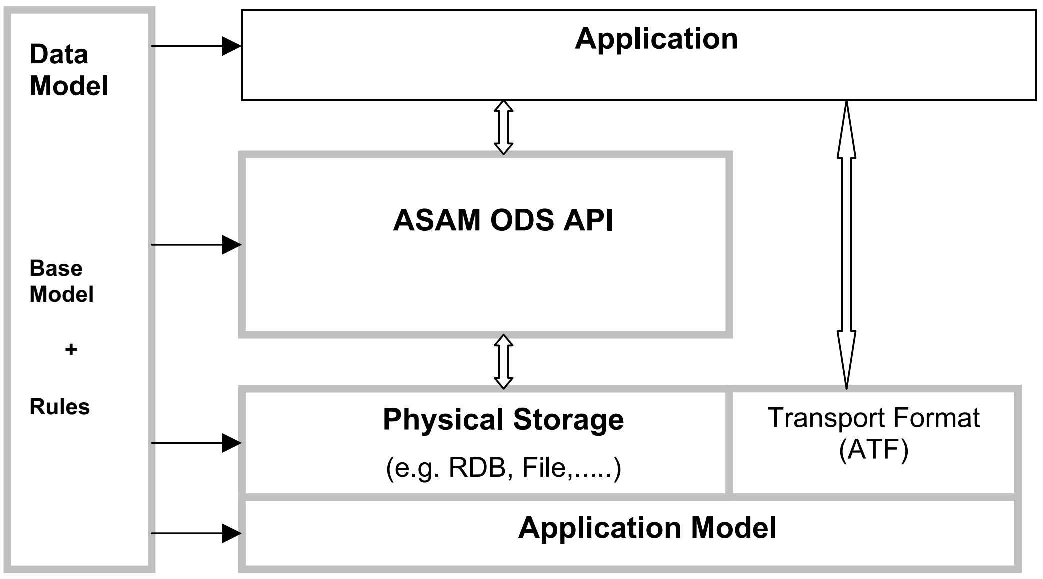
2 ASAM ODS ARCHITECTURE

2.1 INTRODUCTION

The specifications of ASAM ODS Version 5.0 are structured into 4 components, which depend on each other. These structures are:

1. The data model (consisting of the base model and the rules for deriving an application model)
2. The physical storage
3. The Application Programmers Interfaces (APIs)
4. The ASAM Transport Format (ATF)

The data model is the basis, all other components are harmonized with the data model. The following simplified figure shows the relations of data model, application, application interfaces, physical storage, transport format and application model.



All specifications depend on the base model and the rules (generic data model).

The concept enables to handle all application models derived from the base model within the whole system. The control is done by meta information (application model and mapping to the base model).

For example, the physical storage for a relational database is specified in a way, that the meta information is stored in fixed specified tables. Also for the transport format (ATF) the storage of the meta information is specified and for the access layer corresponding methods are defined.

Since the data model is the basis of all specifications, the elements of this model are described within this chapter. The specification of the data model includes the definitions for

the base model

the rules for deriving a project-specific application model

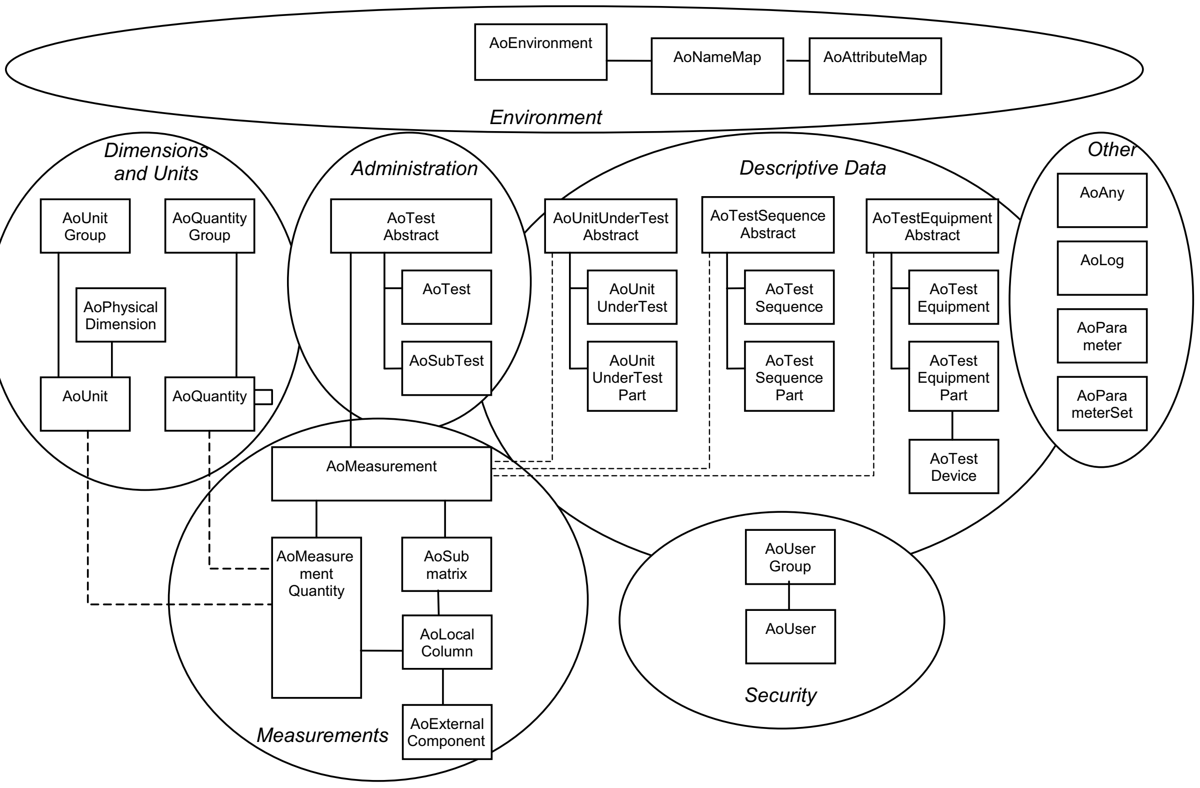
the rules for building instances from the application model.

Base Model → Application Model → Instances

2.2 ELEMENTS OF THE BASE MODEL

2.2.1 OVERVIEW

The following figure shows the ASAM ODS base model:



The base model incorporates the following base elements, from which the application elements may be derived. These elements are grouped according to their usage within automation and measurement systems. A detailed description of each base element is contained in chapter 4.

2.2.2 ENVIRONMENT

The following elements are used to describe the environment of an ASAM ODS server and data storage:

AoEnvironment

AoNameMap

AoAttributeMap

2.2.3 DIMENSIONS AND UNITS

The following elements are used to build a catalog of units and define the relation to SI units (via AoPhysicalDimension):

AoUnit	Units
AoQuantity	Quantities and their properties
AoPhysicalDimension	Physical Dimensions
AoUnitGroup	Unit groups
AoQuantityGroup	Quantity groups

2.2.4 ADMINISTRATION

The following elements are used to build a tree structured administration of measurements and acquired (or evaluated) data, enabling several levels of subtests:

AoTest (derived from AoTestAbstract)

AoSubtest (derived from AoTestAbstract)

2.2.5 MEASUREMENTS

The following elements are used to build structures to store measurement (and evaluation) results:

AoMeasurement	Measurements as bundle of measurement quantities and arrays of AoLocalColumn
AoMeasurementQuantity	Measurement quantities in measurements
AoSubmatrix	Array of LocalColumn
AoLocalColumn	Measurement vector
AoExternalComponent	Reference to a measurement vector in external files

2.2.6 DESCRIPTIVE DATA

The following elements are used to build structures for supplementary descriptions of the tests:

for the Tested Object

AoUnitUnderTest (derived from AoUnitUnderTestAbstract)

AoUnitUnderTestPart (derived from AoUnitUnderTestAbstract)

for the Test Sequence

AoTestSequence (derived from AoTestSequenceAbstract))

AoTestSequencePart (derived from AoTestSequenceAbstract)

for the Test Equipment

AoTestEquipment (derived from AoTestEquipmentAbstract)

AoTestEquipmentPart (derived from AoTestEquipmentAbstract)

AoTestDevice (derived from AoTestEquipmentPart)

2.2.7 SECURITY

The following elements are provided for the storage of security data (which allows to restrict the access to the content of an ASAM ODS server):

AoUser

AoUserGroup

2.2.8 OTHER DATA

The following elements are provided for the storage of other data:

AoAny

AoLog

AoParameter

AoParameterSet

2.3 RULES FOR THE APPLICATION MODEL

To derive an application model from the base model several rules are to be followed. There are rules for deriving application elements from base elements, for naming the derived elements and for including additional attributes.

2.3.1 NAMES OF APPLICATION ELEMENTS

	Base Model	→	Application Model
e.g.:	AoMeasurement	→	EnduranceTest

The derived application elements may be named according to the project-specific requirements. The relation to the base model must be maintained and is stored within the meta information. The names of application elements must never begin with “Ao” (this is reserved for base element names).

2.3.2 BUILDING THE APPLICATION MODEL

There are different rules for deriving one or more application elements from the base model. Additional specifications control which relations are allowed or even prescribed. Please note that there are 3 groups of base elements for which different derivation rules are provided:

- Single derivation
- Tree structure at instance level
- Tree structure at application level.

SINGLE DERIVATION

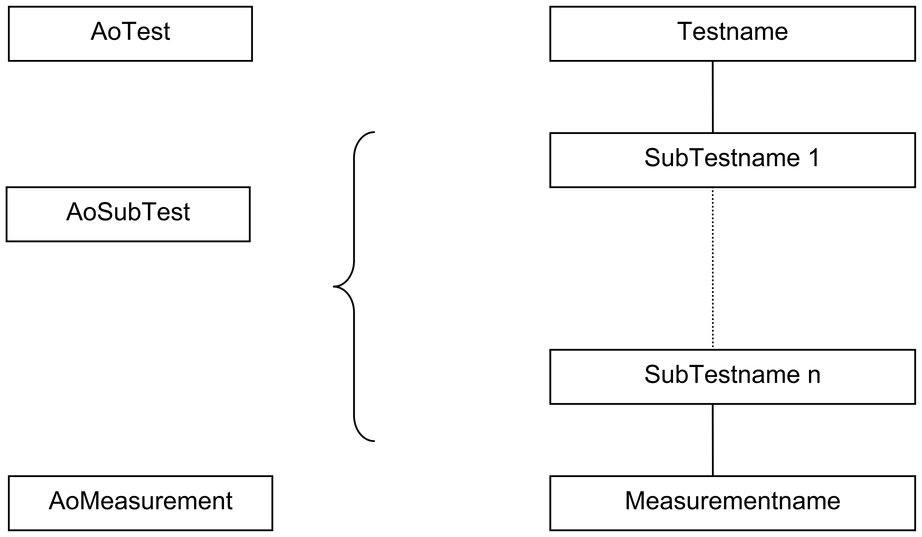
From the following base elements **only one application element** may be derived. These elements are the most precisely specified base elements (through base attributes) within the base model:

AoEnvironment
AoMeasurement
AoMeasurementQuantity
AoPhysicalDimension
AoQuantity
AoQuantityGroup
AoTest
AoUnit
AoUnitGroup
AoUser
AoUserGroup
AoSubmatrix
AoLocalColumn

TREE STRUCTURE AT INSTANCE LEVEL

The following base elements may be used to allow a **tree structure** at the **instance level**

Base Model → **Application Model**



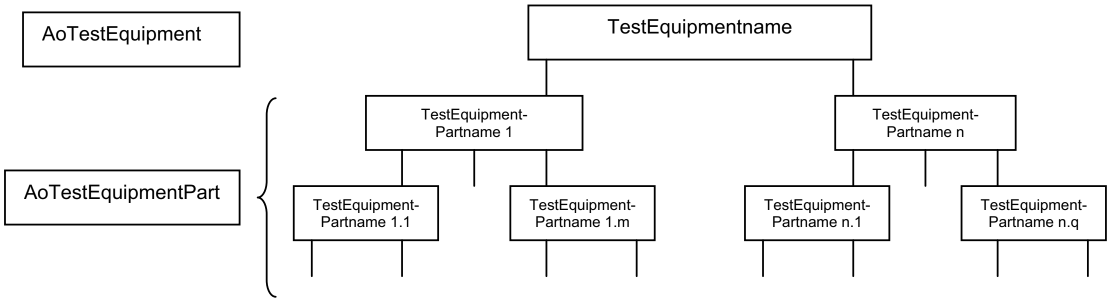
From AoTest exactly one application element may be derived, and from AoSubtest exactly one application element per hierarchical level (1...n) may be derived. On instance level the tree structure is built by multiple instantiation of an application element. As a special case it is allowed to have no application elements of the type AoSubtest. The application element derived from AoMeasurement is only appended to the lowest element of the “test” family. This makes it possible to get a tree-structured administration of measurements (AoMeasurement) at the instance level.

TREE STRUCTURE AT APPLICATION LEVEL

With the following base elements tree structures may be built on application level.

- **For the Test Equipment**

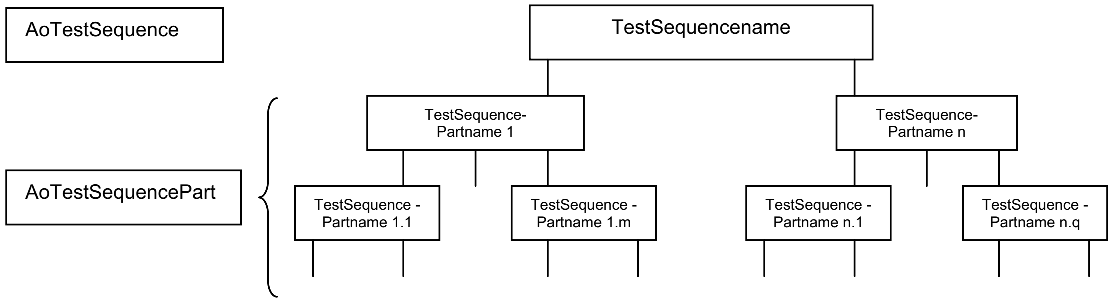
Base Model → **Application Model**


AoTestEquipment
AoTestEquipmentPart
TestEquipmentname
TestEquipment-Partname 1
TestEquipment-Partname n
TestEquipment-Partname 1.1
TestEquipment-Partname 1.m
TestEquipment-Partname n.1
TestEquipment-Partname n.q

This applies also to AoTestDevice (for the hierarchical arrangement of test devices).

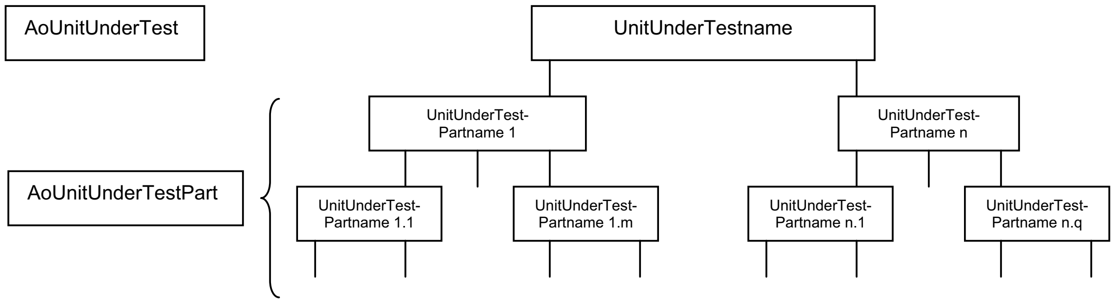
- **For the Test Sequence**

Base Model → **Application Model**


AoTestSequence
AoTestSequencePart
TestSequencename
TestSequence-Partname 1
TestSequence-Partname n
TestSequence - Partname 1.1
TestSequence - Partname 1.m
TestSequence - Partname n.1
TestSequence - Partname n.q

- **For the Test Unit**

Base Model → **Application Model**


AoUnitUnderTest
AoUnitUnderTestPart
UnitUnderTestname
UnitUnderTest-Partname 1
UnitUnderTest-Partname n
UnitUnderTest-Partname 1.1
UnitUnderTest-Partname 1.m
UnitUnderTest-Partname n.1
UnitUnderTest-Partname n.q

In these cases it is allowed to build a tree structure already at the application level by deriving several application elements from AoTestEquipmentPart, AoTestSequencePart and AoUnitUnderTestPart. Additionally, from AoTestEquipment, AoTestSequence and AoUnitUnderTest as many application elements as needed may also be derived. In this way it is possible to define as many tree structures at application level as appropriate.

- **For Application Elements derived from AoAny**

The base element AoAny may be derived as often as needed. In connection with application references arbitrary application models may be built.

2.3.3 MAPPING BASE MODEL AND APPLICATION MODEL

The storage of the alternative names (e.g for different languages) takes place within the two entities **“AoNameMap”** and **“AoAttributeMap”**.

AoNameMap

Within the instances of “AoNameMap” all entities of the application model (the “application elements”) are stored together with their relation to the corresponding base entities. It is possible to build a list of alias names. Additionally a list with the “attribute maps” for this application element is stored.

AoAttributeMap

Each instance of "AoAttributeMap" contains one attribute of an application element respectively. If an attribute was derived from a base attribute, the corresponding relation is also stored. It is further possible to build a list of alias names and to define a unit and a relation to a quantity.

AoEnvironment

Of this entity only one instance should appear in any data set. It is used as bracket for the AoNameMaps (which in turn work as bracket for the AoAttributeMaps) and to store some global statements:

- Number of hierarchy levels for the test,
- Meaning of the alternatives within the alias names.

2.4 ATTRIBUTES IN ODS MODELS

2.4.1 RULES FOR ATTRIBUTES

Analogous to elements there are rules for the attributes at application and instance level. Due to the wide range of application areas it is not possible to specify all attributes within the base model. Therefore the following rules control the naming of base attributes and the addition of application-specific attributes.

At the transition from one model level to the next one attributes may be added.

Base Model	→	**Application Model**	→	**Instances**
Base attributes		Base attributes		Base attributes
		Application attributes		Application attributes
				Instance attributes

Remarks

1. The names of the base attributes may be changed within the application model. This information is stored within the meta information.
2. Project-specific application attributes may be added within the application model. Also this information is stored within the meta information.
3. All application attribute names must be stored within the meta information.
4. Attributes may also be added to any instance. In this case attribute name, type and value are required. They are stored within the data.

2.4.2 ATTRIBUTES OF ALL ELEMENTS

All base elements (and therefore also all application elements) possess the following main base attributes:

id	Unique ID for the instances of an application element
name	Name of the instance, only unique within its predecessor in a hierarchy
description	Describing text for the instance (optional)
version	Version of the instance (optional)
version_date	Date of the version change (optional)

Most of the other base attributes control the relations between the base elements. A base attribute is automatically passed down to all the application elements derived from that base element.

Each Base element has a (ASAM ODS defined) base ID (BID) which uniquely allows to identify the base element without the need to compare strings (names).

In the following sections the base elements, their BIDs, and their base attributes are described.

2.5 DATA TYPE USAGE IN ASAM ODS

2.5.1 GENERAL NOTES TO DATA TYPES IN ASAM ODS

Data types appear nearly everywhere within the ASAM ODS specification chapters. Special care must be taken to understand and correctly use them.

First of all, only basic data types will be considered in this section (‘data type’ thus will exactly mean ‘basic data type’). However distinguishing between basic types and non-basic types is not easy.
From an object-oriented point of view, every class or interface can be considered to be data type. This section will not treat classes or interfaces as basic data types.
From a data modeling point of view each data structure (even the most complex ones) can be considered to be a data type. Again, this section will treat them as basic data types only if their structure is quite simple (thereby knowing that this is a quite vague definition of ‘basic’).

ASAM ODS clearly distinguishes between a data type and its number.

Each data type is given a <u>name</u> (e.g. T_SHORT) and a <u>type definition</u>. The type definition relates the data type to the a priori known primitive types. What is a primitive type depends on the specific context. Defining data types for a Java context requires to relate them to the predefined Java data types, defining them for a CORBA context requires to relate them to the CORBA predefined data types (which themselves are further related to language dependent primitive data types through the CORBA language binding), defining them for a STEP EXPRESS context requires to relate them to the predefined STEP EXPRESS data types, and defining them for a RPC context requires to relate them to the RPC predefined data types, etc.. That is why the data type definitions will differ in the subsequent chapters of the ASAM ODS specification.

Most of the data types are given a <u>unique number</u> (within an enumeration of the available data types) so that it can be referenced at run time by its number. This allows to tell about the types of data that are exchanged through an interface or the types of data that are contained in a physical storage (database, file) at run time. The data type of the measurement values (of e.g. engine speed) that are kept in the data store thus do not need to be specified in advance for all future cases. Instead the application producing them may decide in each case about the preferred data type; its number will be delivered at the API or to the physical storage together with the data and the receiving application will be able to treat the data appropriately according to their current data type.
The <u>numbers</u> are also given a <u>name</u> each (e.g. DT_SHORT); those names should not be confused with the names of the data types (though there is a 1:1 relationship between them). Names for data type numbers ease the understanding of enumerations used in switch-case-constructs; instead of ‘case 2:’ one would rather read ‘case_DT_SHORT’. Names for data type numbers are also often presented to the user interfaces; people therefore tend to talk of ‘the data type DT_SHORT’ (which, strictly speaking, is not correct, but tolerable).

Data types are defined in

➢ the base model (see chapter 4 for details). These are the most general and implementation independent types and are used to specify application models. They are defined by an appropriate ENTITY construct of the STEP EXPRESS notation and thereby related to primitive types of STEP EXPRESS.

- the RPC-API (see chapter 9 for details). Not all of the data types defined by the base model may be used in the RPC-API due to implementation restrictions. This may have a major impact on the visibility and accessibility of data through the RPC-API. Thus the RPC-API should only be used when the application model does not contain unsupported data types. On the other hand, when designing an application model that will be accessed through the RPC-API no unsupported data types should be used.
- the OO-API (see chapter 10 for details). This API is much more flexible. Besides the data types defined in the base model it defines and uses additional data types. Most of these additional types are sequences of the base model types (or even sequences of sequences) and thus allow an easier and more generic programming to access data.
- the ATF/CLA (see chapter 5 for details). In order to store not only the information itself but also the data type with which it is placed in the ATF file, ATF/CLA specifies a keyword for each of the data types used. These keywords match the names of the data type numbers (the enumeration names!) of all data types defined in the base model.
- the ATF/XML (see chapter 6 for details). In order to store not only the information itself but also the data type with which it is placed in the ATF file, ATF/XML specifies a keyword for each of the data types used. These keywords match the names of the data type numbers (the enumeration names!) of all data types defined in the base model, but also contain several data types that are only used in the OO-API.
- the physical storage specification (see chapter 3 for details). The physical storage provides a specification of tables in a relational database that finally shall hold the information on tests, measurements, etc.. The data types of the information stored however is usually not predefined. Thus the database must not only store the information but also store the data type that is used. For this purpose, only the data type numbers are stored. The physical storage allows to specify all basic data types that are defined in the base model, and the first order sequence data types that are specified in the OO-API.
- the application models that are already specified or will come up in the future (see e.g. chapters 11-13 for details). Data types used here must match the data types that are defined in the base model. Exceptions are explained in section 2.5.3 and 2.5.4.

Names of data types are written in uppercase. Names of the data type enumerations are also given in uppercase letters throughout the ASAM ODS specification.

Data type numbers are always the same if they stand for the same data type (the enumeration is defined in the same order throughout the ASAM ODS specification). The mapping of data types to their physical representation should be identical within any context. This means that e.g. T_SHORT should be mapped to a 16 bit Integer data type no matter whether it is implemented in Java, C++, Pascal, or whatever language and platform is used.

In the subsequent sections special aspects of data type usage are further explained.

Section 2.5.2 describes the relationship between the API data types and the base model data types.

Section 2.5.3 describes, in which cases the data types of application attributes may differ from the data types of the corresponding base attributes.

Section 2.5.4 explains how base relations (which can be seen as special attributes) may be modified regarding the relation range when defining an application model.

Thus an ASAM ODS server and client have to be aware that attributes may have a different data type than defined in the base model.

The data types of all attributes that are not explicitly mentioned in these sections must match the default data types defined in the base model.

Section 2.5.5 finally explains how some ASAM-wide standardized data types match the ASAM ODS data types and how ASAM ODS will use these ASAM data types in the future.

2.5.2 DATA TYPES OF THE BASE MODEL AND THE APIS

The names of the data types in the base model and the APIs as well as the names of the data type enumerations are not always identical. The following table compares data type names and their numbers of the base model with those in the APIs.

OO-API		RPC-API		Base model		
data type	enumeration	data type	enumeration	data type	enumeration	
--	DT_UNKNOWN (=0)	--	DT_RESERVED (=0)	--	DT_UNKNOWN (=0)	
T_STRING	DT_STRING (=1)	char[]	DT_STRING (=1)	t_string	DT_STRING (=1)	
T_SHORT	DT_SHORT (=2)	short	DT_SHORT (=2)	t_short	DT_SHORT (=2)	
T_FLOAT	DT_FLOAT (=3)	float	DT_FLOAT (=3)	t_float	DT_FLOAT (=3)	
T_BOOLEAN	DT_BOOLEAN (=4)			t_boolean	DT_BOOLEAN (=4)	
T_BYTE	DT_BYTE (=5)	unsigned char	DT_BYTE (=5)	t_byte	DT_BYTE (=5)	
T_LONG	DT_LONG (=6)	long	DT_LONG (=6)	t_long	DT_LONG (=6)	
T_DOUBLE	DT_DOUBLE (=7)	double	DT_DOUBLE (=7)	t_double	DT_DOUBLE (=7)	
T_LONGLONG	DT_LONGLONG (=8)			t_longlong	DT_LONGLONG (=8)	
--	DT_ID (=9)			--	DT_ID (=9)	
T_DATE	DT_DATE (=10)	char[]	DT_DATE (=10)	t_date	DT_DATE (=10)	
T_BYTESTR	DT_BYTESTR (=11)	bstream	DT_BYTESTR (=11)	t_bytestr	DT_BYTESTR (=11)	
T_BLOB	DT_BLOB (=12)	Blob	DT_BLOB (=12)	t_blob	DT_BLOB (=12)	
T_COMPLEX	DT_COMPLEX (=13)			t_complex	DT_COMPLEX (=13)	
T_DCOMPLEX	DT_DCOMPLEX (=14)			t_dcomplex	DT_DCOMPLEX (=14)	
S_STRING	DS_STRING (=15)					
S_SHORT	DS_SHORT (=16)					b)
S_FLOAT	DS_FLOAT (=17)					b)
S_BOOLEAN	DS_BOOLEAN (=18)					b)
S_BYTE	DS_BYTE (=19)					b)
S_LONG	DS_LONG (=20)					b)
S_DOUBLE	DS_DOUBLE (=21)					b)

OO-API		RPC-API		Base model		
data type	enumeration	data type	enumeration	data type	enumeration	
S_LONGLONG	DS_LONGLONG (=22)					a)
S_COMPLEX	DS_COMPLEX (=23)					c)
S_DCOMPLEX	DS_DCOMPLEX (=24)					c)
--	DS_ID (=25)					
S_DATE	DS_DATE (=26)					a)
S_BYTESTR	DS_BYTESTR (=27)					b)
T_ExternalReference	DT_EXTERNALREFERENCE (=28)			t_externalreference	DT_EXTERNAL_REFERENCE (=28)	
S_ExternalReference	DS_EXTERNALREFERENCE (=29)					a)
T_LONG	DT_ENUM (=30)				DT_ENUM (=30)	
S_LONG	DS_ENUM (=31)					

Sequence data types of the OO-API may be represented in STEP EXPRESS as

LIST [0:?] OF ### (in cases a) above)
LIST [1:?] OF ### (in cases b) above)
LIST [2:?] OF ### (in cases c) above)

where ### denotes an appropriate basic data type.

2.5.3 ALTERNATIVES TO DATA TYPES OF BASE ATTRIBUTES

In the base model, all base attributes have strictly defined data types. Normally the application attributes derived from the base attributes must have the same data type. There are some data types which are compatible, so ASAM ODS will in some cases allow the designer of the application model to overload the data type of the base attributes.

The following base attributes can be used with the given allowed alternative data type, any other data type conversion can cause an error. The server and the client must not support any other data type conversion for any base attribute. It is up to the application model designer to decide which data type will be used.

Base attribute name	Default data type enum name	Allowed alternative
external_references	DS_EXTERNALREFERENCE	DT_EXTERNALREFERENCE
dimension	DS_LONG	DT_LONG

For the following base attributes, the server can deliver the accepted alternative. These accepted alternatives are available in existing and installed application models and servers. ***These accepted alternatives cannot be used in new designed application models.***

Base attribute name	Default data type enum name	Allowed alternative
id	DT_LONGLONG	DT_LONG
factor	DT_DOUBLE	DT_FLOAT
offset	DT_DOUBLE	DT_FLOAT
superuser_flag	DT_SHORT	DT_BYTE
version	DT_STRING	DT_LONG

2.5.4 ALTERNATIVES TO RELATION RANGES OF BASE RELATIONS

The base relations are special base attributes; they can be treated as a set of attributes. Each of those attributes (each relation) also has an inverse relation.

There are information relations which are defined in the base model as N:M relations. It is allowed to reduce these relations from an N:M relation to a 1:N relation.

2.5.5 THE STANDARDIZED ASAM DATA TYPES

In 2000, the ASAM e.V. started a project to harmonize the data types used in different working areas (ACI, GDI, MCD, ODS etc.).

This lead to a harmonization paper officially released in 2002, which compared the existing ASAM ODS data types with the ones defined for usage throughout the whole ASAM, and specified ASAM wide standardized data types (the 'ASAM data types'). Detailed information on the complete set may be requested from the ASAM e.V.. This section compares the ASAM ODS data types with the ASAM data types and explains the migration path towards the ASAM data types.

MAPPING BETWEEN ASAM ODS AND ASAM DATA TYPES

The following table shows those ASAM ODS data types that have an identical ASAM data type counterpart (only the name of the data type is different); these may be used without touching programming logic by simply renaming them; they are binary compatible to each other. Whoever starts to implement ASAM ODS servers or clients may use these ASAM data types instead of the ASAM ODS data types.

ASAM ODS data type	ASAM data type	Description
T_STRING	A_ASCIISTRING	zero-terminated character field; ASCII coded (ISO-8859-1(Latin-1)) maximum length 2^{32}-1 plus delimiter 0x00
T_SHORT	A_INT16	signed integer (16 bit)
T_FLOAT	A_FLOAT32	float, IEEE 754 single precision (32 bit)
T_BOOLEAN	A_BOOLEAN	boolean (8 bit); 0x00=FALSE, any other value means TRUE
T_BYTE	A_INT8	signed integer (8 bit)
T_LONG	A_INT32	signed integer (32 bit)
T_DOUBLE	A_FLOAT64	float, IEEE 754 double precision (64 bit)
T_LONGLONG	A_INT64	signed integer (64 bit)
T_COMPLEX	A_COMPLEX32	complex; structure (2x32 bit); consisting of two elements of single precision float values (see above), one giving the real, the other giving the imaginary part.
T_DCOMPLEX	A_COMPLEX64	complex; structure (2x64 bit); consisting of two elements of double precision float values (see above), one giving the real, the other giving the imaginary part.
T_LONG	A_ENUM	enumeration

Remark: ASAM ODS did not specify a separate data type for enumerations but used T_LONG instead. However to distinguish enumerations from numeric information a specific data type number was given: DT_ENUM (=30).

Some of the ASAM ODS data types differ from the ASAM data types:

Date and time: The ASAM ODS data type T_DATE is specified as zero-terminated string with a specific semantic: the date and time components year (YYYY), month (MM), day (DD), hour (hh), minute (mm), second (ss), millisecond (lll), microsecond (ccc), nanosecond (nnn) etc. are coded as "YYYYMMDDhhmmsslllcccnnn...", terminated by 0x00. Starting from the left, as many components as required may be used, the other may be omitted. Thus the minimum information is YYYY.

An ASAM data type which comes close to this specification is A_TIME_STRUCT. It is a binary representation of date and time and is defined as:

```
struct A_TIME_STRUCT
 {
 A_INT16  n16Year;
 A_INT8   n8Month;
 A_INT8   n8DayOfMonth;
 A_INT8   n8Hour;
 A_INT8   n8Minute;
 A_INT8   n8Second;
 A_INT16  n16MilliSecond;
 A_INT16  n16MicroSecond;
 A_INT16  n16NanoSecond;
 A_INT32  n32TimeZoneDiff; // Time difference between UTC and local time in
                           // seconds(-13 hours to 13 hours)
 };
```

Conversion methods must be used that allow to translate one representation into the other.

Binary data streams appear in ASAM ODS in two data types:

T_BYTESTR: is a sequence of bytes; the way the length information is stored and interpreted depends on the specific context (CORBA, RPC, ...).

T_BLOB: is a zero-terminated string (that may contain a header or other information), followed by a sequence of bytes; as above, the way the length information is stored and interpreted depends on the specific context.

Among the ASAM data types A_BYTEFIELD is thought to map to them. It is defined as

```
struct A_BYTEFIELD
{
  A_UINT32 un32Bytes ;   // number of bytes in bytefield [1 .. 2^32 - 1
  A_UINT8  aun8Field[un32Bytes];
};
```

Conversion methods must be used that allow to translate one representation into the other.

Other ASAM ODS data types like T_EXTERNALREFERENCE (which consists of three strings) and all sequence data types do not have a counterpart in the list of ASAM data types. Therefore these ASAM ODS data types will be further used as they are.

The **enumerations** 0..29 are defined in the ASAM data type specification as they are defined in ASAM ODS.

MIGRATION PATH

In many parts of the ASAM ODS specification the names of the data type enumeration elements (e.g. DT_SHORT) are used, which are harmonized ASAM-wide. This is the case for ATF/CLA, ATF/XML, the physical storage, and the application models for specific application areas.

As ASAM ODS has a long history with lots of implementations worldwide, it turned out that it is nearly impossible to change the data types within the existing implementations. Therefore, ASAM ODS decided to further support the ASAM ODS data types within the OO-API and the RPC data type primitives that are specified by the ONC and that are directly used by the RPC-API.

At some point of time in the future, the base model may either switch to the ASAM data types or support both.

Future ASAM ODS specifications will build upon the ASAM data types and thus allow to write applications that combine a variety of ASAM interfaces more easily.

2.6 THE APPLICATION INTERFACES

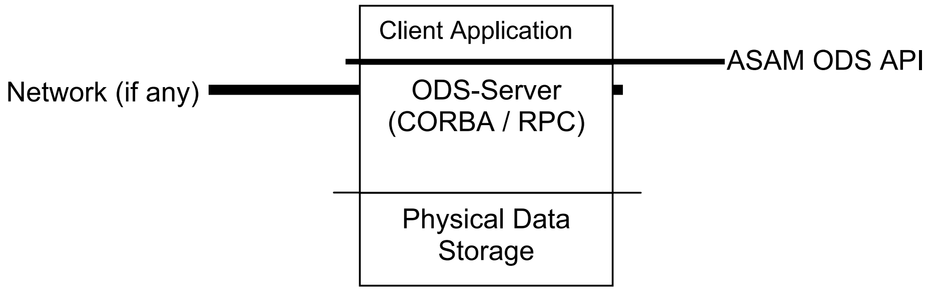
2.6.1 BASIC INFORMATION

ASAM ODS provides Application Interfaces (APIs) to give any application a standardized and effectively designed access to the data.

LOCATING THE APPLICATION INTERFACE (API) IN ASAM ODS

An analysis of the present data formats for measurement and automation systems shows that unification to one fixed format is not possible; however, there is a common ground. In ASAM ODS this common ground is supported by the **base model**. This base model is a *general, application independent, logical data model*, already clearly defining the structure of data to be processed within ASAM ODS, but it only roughly outlines the information to be stored.

In addition to the base model, **methods** are defined in ASAM ODS that help an application to create from the base model an *application specific data model*. This **application model** describes the complete data model of the application. Other methods, also provided by ASAM ODS, are used to access the **instances,** that is the data stored in the application model.



Location of the API

As the figure shows, an application accesses ASAM ODS data solely by calling the functions of the ASAM ODS API. The API hides any network transfer; in cases where the application and the physical data storage reside on different systems, either the CORBA (when using the OO-API) or the RPC (when using the RPC-API) technology cares for the correct translation of the client-side requests into server-side requests and the transport of the response data.

The API provides the hardware-independent and data-format(e.g. file or database)-independent part of ASAM ODS; all API data accesses will be forwarded by the ASAM ODS API to the data servers. The ASAM ODS API abstracts, in the course of this, from the hardware specific representation of the ASAM data types (e.g. 4-byte-Integer) on the different computer platforms. The ASAM ODS API exchanges the data from the server specific representation into a client specific representation; the informational contents and the structure will not be changed by the ASAM ODS API. The data server, at last, abstracts the data format and the physically used data types; this means, it converts data in an ASAM conform structure and in ASAM internal types.

ODS servers implement the actual access routines to data. Typically, the ASAM transport format (ATF) and the ASAM data base format (physical storage) will be supported, but in addition to this, also existing formats - foreign formats - ought to be supported as long as they can be provided to the client through the ASAM ODS API. The purpose of the API and its interfaces to the application will be described in the next sections.

TASK OF THE API AND GENERAL CONSIDERATIONS

The API defined in this concept provides functions to support the application.

There are functions to create the application model according to the rules of the base model. Some basic information about the application model will be stored with the data by ASAM ODS in the **environment**. If this environment is accessed, this information may be used to set up the application model automatically.

Functions for writing, reading and changing data are also part of the API.

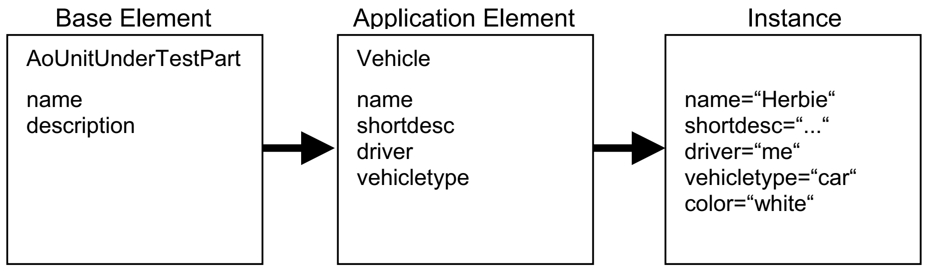
The API operates with objects subdivided mainly of the following three groups:

- base elements
- application elements
- instance elements

Base elements and **application elements** define the data types to be processed by the application. The base elements are the data structures defined in ASAM ODS which build up the base model. They define all relations between the individual data objects supported by ASAM ODS. Also, for each base element they define a number of application independent **base attributes** which are contained in every application element derived from that base element. The application is able to add further application specific attributes to each application element.
All information about the application model is encoded in the application elements. The data to be processed by the application will be stored in instances of these application elements as attribute values. An instance is able to contain own, instance specific attributes besides attributes clearly defined by the corresponding application element.

EXAMPLE: **BASE ELEMENT, APPLICATION ELEMENT, AND INSTANCE**

The following figure shows the base element with two of its base attributes. From it, an application element 'Vehicle' has been built which uses the base attributes 'name' (unchanged) and 'description' (now named 'shortdesc'), and adds two more attributes (application attributes) 'driver' and 'vehicletype'. Neither the base element nor the application element holds values for the attributes. Instead, for that purpose instances of the application element 'Vehicle' are built. One of the instances is shown; it contains a value for each of the application attributes and besides them, it adds another attribute (an instance attribute) named 'color' with a corresponding value. This instance attribute is specific to the very instance; other instances may miss it.



The interface towards the application, the **Application Programming Interface (API)**, provides the methods for creating, storing and reading an application model, and the instances. The methods for storing and reading of data stored in these models are also a part of this interface. The base model is implicitly or explicitly known by the server and may only be read.

For an ASAM ODS application, running on a certain computer, it ought to be possible to access data stored on other computers or even other platforms. Therefore, a client-server architecture is used.

REQUIREMENTS GIVEN BY THE DATA MODEL

ASAM ODS defines the base model; from it the application model has to be built up. During definition of a new application model, the ASAM ODS server has to guarantee the accordance with the base model and its rules. While reading, a stored application model has to be rebuilt with lowest effort (for the application).

As already described above (for technical reasons) not all data of one measurement will be stored together as a complete matrix; it will be distributed into different submatrices. For the application however, an easy access to the complete data needs to be provided. The application usually does not care for data storage efficiency. That is why the value matrix is provided to the application; it needs to be combined out of the single submatrices during runtime by the ASAM ODS server automatically.

Part of the base model contains information that allow very versatile mechanisms for an (optional) access control to ASAM ODS objects.

APPLICATION-BASED API REQUIREMENTS AND TYPICAL USAGE

There are a number of different application types which have partly different requirements to the API. The most common application types will be characterized by their typical actions:

- *Measurement definition*: Set-up of a new application model; for each environment this has to be done only once.
- *Measurement preparation*: Creating test descriptions, creating pattern structures, changing test descriptions.
- *Measurement data recording*: Writing measurement data.
- *Visualization of data*: Reading all measured data of a measurement quantity, reading attributes.
- *Analysis of data*: Reading measurement data; creating new data, sequential reading; reading implicit time information, reading on a given time grid like t0-Δt-grid (e.g. mathematical processes).
- *Preparation of data*: Reading selected measurement data or groups of attributes to be further processed (sorting, compressing, calibrating, etc.).
- *Editing of data*: Reading and changing single measured values (e.g. data editor).
- *Converter for data stocks*: Reading or writing all data (once).

Data recording during a measurement is often a time-critical process where the measurement values have to be stored within a very short time period. Even though it is not the (main) purpose of ASAM ODS to support those time-critical operations, ASAM ODS can be used within its limitations for measurement recording.

Furtheron most applications have to be able to process an existing application model; which means to determine the set-up of the application model from the API, to navigate within it and to find the required data.

These different actions of the specific application types have the following requirements to the functionality of the API:

1. Processing the application model
   - Navigating through the application model.
2. Processing the data
   - Sequentially reading all data of one or several measurement quantities; in this case, the processing time is most important.
   - Reading selected measurement data; here, the comfort of the selection criteria takes the first place but the processing time also plays an important role.
   - Sequentially writing measurement data.
   - Random access writing or changing single measured data; as the value matrix is (usually) combined of several submatrices, changing measurement values is only possible under some restrictions.
   - Reading attributes; single ones as well as in groups.
   - Writing or changing attributes.

REQUIREMENTS GIVEN BY THE PHYSICAL STORAGE

The requirements by a physical storage are actually requirements by the data server supporting the corresponding file format on the corresponding platform. The following requirements ought to be named:

- Not all data servers can write data.
- Data servers, especially foreign format servers, do often not support all data types provided in ASAM ODS for writing; during reading, of course, only those data types have to be supported which are needed for the data that is stored.

GENERAL REQUIREMENTS

Besides the previous requirements, there are several general requirements:

- The API functions have to be machine-independent.
- The calls of the API functions have to be independent from their implementation.
- The functions are not permitted to assume that the data to be processed can be stored completely in their memory. Portioning, especially during sorting and joining, often leads to huge time delays and, therefore, it ought to be avoided.
- The set-up of the value matrix and the views is partly very complex and time-consuming (e.g. join, merge or sort), enough information needs to be stored in memory so that the set-up has not to be performed at each data access.
- A version management needs to be integrated to consider the version of the base model from which the current application model is derived.

2.6.2 WORKING WITH THE APPLICATION INTERFACE

In this section, the main functionality of the application interface will further be described.

Besides processing the application model, management and evaluation of access rights, the management of all data relevant for a measurement is the main function of ASAM ODS. The information to be handled during a measurement can be quite different, therefore, it is separated into the two categories **measurement data** and **attributes:**

- In measurement data, all data is combined which consists of a complete data vector (often a time flow) for usually one measurement quantity; this category includes the actual measurement results, the reference quantities, and processed data (like the results of mathematical calculations).
- The term attributes covers all information in the framework of a measurement, e.g. information on the measured quantities (names, units,...), the unit under test, the test equipment used, etc.. This type of data usually consists of a small number of values.

The most important functions of the API can be divided into four fields:

- Processing the application model
- Managing the measurement data
- Managing the attributes

Before explaining these requirements for the API, some general information on how to access the ASAM ODS data will be described. For details please refer to chapters 9 and 10.

ACCESSING ASAM ODS DATA

To access an existing application model or its attributes, the application needs to connect to an ASAM ODS server. Depending on which API and which underlying distributed technology is used, the details of this connection process may differ.

Using the OO-API with CORBA typically requires to connect to the CORBA name service and ask for an ASAM ODS factory object (AoFactory); this allows to create a new session with the ASAM ODS server, which then provides access to the base model, application model, and instances.

Using the RPC-API, a client has to be created (providing a node name and a RPC number). This client will establish a session (which is called environment in that context) with the server; the env_Id exchanged during that process is further used by the client to access the application model and the instances.

PROCESSING THE APPLICATION MODEL

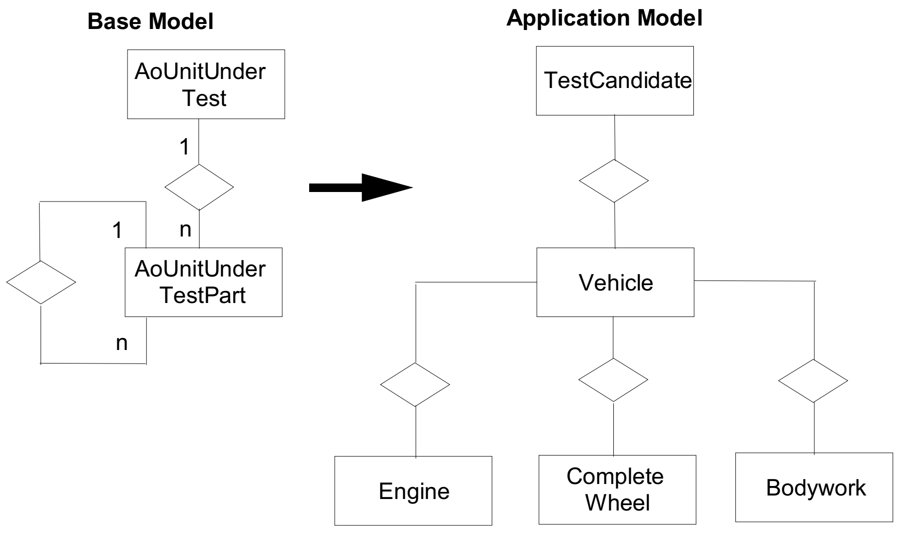
The application model is a logical, application specific data model to provide the description of the application data managed by ASAM ODS. Processing the application model by different applications may be divided into two sections: creating or modifying application models, and navigating through an application model.

Creating / modifying application models

The example below shows a part of the base model and a possible application model derived from it. Two important aspects of the ASAM ODS architecture are shown, to clarify the creation of an application model:

- The *allocation of application specific names* for application elements makes it possible to use an application specific terminology instead of the general terminology of the base model.
- By the *creation of hierarchies* from a recursively defined base element (e.g. the unit under test part), a complete tree of application elements can be created.

EXAMPLE: **CREATING AN APPLICATION MODEL**



A third aspect of ASAM ODS in the context of defining the application model is not contained in this example (see example in previous section):

- The *definition of application specific attributes*, which enable a more specific description of the application elements.

Navigating through the application model

Some applications, especially those for evaluating data, have to be able to operate with any application model. Therefore, methods are provided with which the application is able to ask for the setup of the model or to navigate within the application model, analogous to a directory tree. This includes listing all children of an application element or all children derived from a certain base element. Also part of it is listing all instances of this application element. For these lists, search patterns for the element or instance names may be specified to restrict the selection.

All elements of an application model have a unique name; this means, an application element is sufficiently qualified by its name.

For processing the application model, the following functionality is provided through the API:

- Finding all application elements
- Finding all attributes of an application element
- Asking for attribute information (data type, etc.)
- Finding all children of an application element (with search pattern)
- Finding all children of an application element which has been derived from a certain base element (with search pattern)
- Finding all instances of an application element (with search pattern)

MANAGING THE MEASUREMENT DATA

The main task of ASAM ODS is to manage the measurement data. According to the ASAM ODS data model, measurement data will be kept within instances of AoLocalColumn. A local column contains data of exactly one measurement quantity. A measurement generally will produce data of more than one measurement quantity; they are put into one local column each. Local columns that contain data being generated similarly (that is which have the same number of data values and where the corresponding data values are all related to e.g. the same time stamp) are related to the same submatrix; submatrices finally are related to the measurement where they were created.

The measurement data have important properties (e.g. a unit, a data type, ...); for this reason, the measurement quantity is provided as an independent application element containing the corresponding attributes. An attribute (the so-called value flags) of the local column contains additional important information about the values; e.g. whether they are valid, modified or should be cut out during visualization. This attribute can either hold a single value to be valid for all values of the measurement quantity or it is inserted as a complete vector with all the information for each value stored in it.

Solely for the convenience of the programmer during reading access, a complete matrix containing all values of all measurement quantities of a measurement will be generated by the ASAM ODS server: the value matrix; the columns of the matrix correspond to the measurement quantities, the rows to e.g. the time stamps (or the sensor locations, ...).

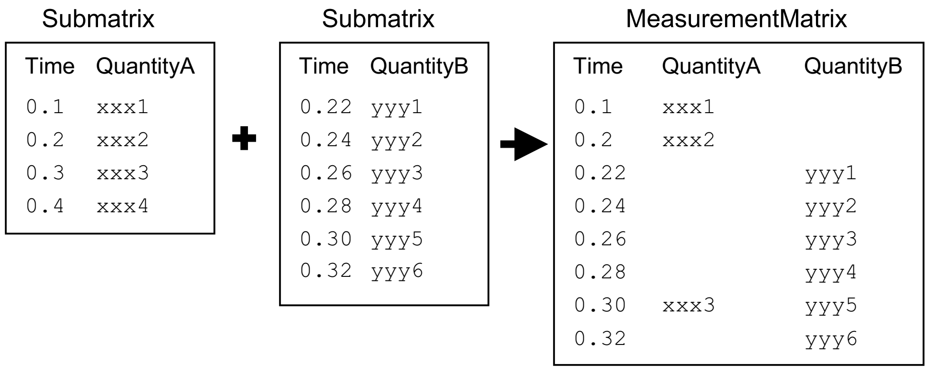
For the data storage only submatrices are used. This is due to the fact that a value matrix will typically contain bigger gaps (undefined values) as not all measurement quantities are measured simultaneously throughout the measurement. Therefore, the value matrix will be divided into parts (the submatrices) so that each of the submatrices does not contain gaps anymore. Submatrices are *homogeneous* and thus require the least possible amount of memory. Only the submatrices of a value matrix are actually regarded during storage; the only additional information required are the rules for set-up of the value matrix from the individual submatrices (the submatrix links).

This type of storage has two advantages:

- The gaps in the value matrix do not require space on the server.
- This separation typically corresponds very well to the physical conditions during a measurement: data is often given as submatrices, because measurement devices used during a test store the measured data locally and not necessarily in the same time resolution as other devices.

EXAMPLE: **SUBMATRICES AND VALUE MATRIX**

In this example, two homogeneous submatrices are combined into one value matrix by merging the time information. The value matrix contains gaps.



Submatrix

Time	QuantityA
0.1	xxx1
0.2	xxx2
0.3	xxx3
0.4	xxx4

Submatrix

Time	QuantityB
0.22	yyy1
0.24	yyy2
0.26	yyy3
0.28	yyy4
0.30	yyy5
0.32	yyy6

MeasurementMatrix

Time	QuantityA	QuantityB
0.1	xxx1	
0.2	xxx2	
0.22		yyy1
0.24		yyy2
0.26		yyy3
0.28		yyy4
0.30	xxx3	yyy5
0.32		yyy6

The access to measurement data is quite different during reading, writing, changing and deleting:

Reading:

- Data is requested from a concrete measurement; the submatrix from which this data is taken will not be named.
- Several measurement quantities may be read at once (if needed).
- The required local columns will be selected by a selection criterion.

Writing (adding of new quantities or new local columns to the measurement):

- Several local columns will be written in parallel.
- The local columns can be stored as new or added to existing submatrices.
- Appending/adding one or more measured points at a given position in a submatrix.
- The submatrix to which the data should be stored, has to be known.

Changing (of existing quantities and existing local columns):

- The values themselves and also the flags of the values, such as visualization and validity, can be changed.
- It will be changed point-wise (one or more measured points out of a submatrix).
- Typically, one local column (resp. measurement quantity) will be changed at a time.
- Only the measurement will be named; the submatrix is internally known.
- A selection criterion can be given.
- Changed data can be stored as a new version of the quantity; old data may be kept.

Deleting

- A complete submatrix may be deleted.
- One or more measured points may be deleted from a submatrix.
- The submatrix from which the local column is taken has to be specified by the client.

During all accesses to data, besides the value also the value flags will be read or written; the value flags cannot be accessed themselves. If all values own the same flags; e.g. all values are valid, visible and modified, the flags will be handled as one scalar for all values; if flags are different for different values, an own set of flags will be inserted for each value, which means, the value flags build up a complete vector.

These different types of accesses result in a quite diverse set of API methods.

MANAGING THE ATTRIBUTES

Attributes can be managed in ASAM ODS for almost all base elements; some elements only have the purpose to manage attributes, e.g. the elements unit under test, unit under test part, test equipment, etc..

The attributes are stored with the corresponding objects in form of name-value pairs. A unit from the unit catalog can be assigned to an attribute.

Processing the predefined attributes

Each instance inherits a number of attributes from the corresponding application element; these are a combination of the base attributes already defined by the base element and the application specific attributes of the corresponding application element. The values of these attributes can be read by the application and changed (if requested). As every instance of an application element can also contain additional instance attributes, an application may want to list all attributes of an instance; for convenience, a search pattern for the attribute names may be used.

The base attributes may be renamed within an application model. An application has to be able to determine the relation between base name and application name of the base attributes. Access to base attributes has to be possible via both names.

Processing of instance specific attributes

Additionally to the pre-defined attributes, an application is able to store further attributes as instance specific attributes in an instance element. Methods are provided for creating and deleting of instance specific attributes. The values of instance attributes will be set directly during creation.

The instance specific attributes are not different from the pre-defined attributes during reading and searching (see next section).

Processing relations

Relations (which may be regarded as a specific kind of attributes to en element) enable to specify how instances link to each other (e.g., the relation between measurement quantity and the corresponding unit).

These relations can be divided into three types:

- **Father-child relation:** Some application elements and their associated instance elements have exactly one father in the hierarchy of the data model. This relation type is used to describe the reference between them.
- **Property relation:** This relation is used if an instance element has a certain property, e.g. as a measurement quantity has a certain unit, in which it will be measured.
- **Informational relation:** Some information is not stored individually with every corresponding instance element. Instead a separate object is created, that can be used several times, e.g. the description of a unit under test or a quantity group. Informational relations typically describe those links.

Which relations are possible is defined in the base model; which ones are actually available will be defined in the application model. The mandatory relations have to be available in each application model.

Relations may be created or deleted by the use of the API.

Search via the attribute values

In many cases, it is not useful to list all instances of an application element; typical requirements are:

- „List all vehicles of model year 1993“
- „List all units which belong to base unit ‘abc’ “
- „List all measurements in which engine ‘xyz’ was involved“

To restrict the result of a method call to a limited number of items, the method to list instances includes a select condition to qualify the requested instances by values of their attributes. It is possible to specify the selection criterion not only by using the attributes of the instances themselves but also attributes of instances of other application elements that are related to this instance.

The provided functionality

The following list is a summary of the most important functionality for processing attributes:

- Reading attribute values
- Writing attribute values
- Creating instance specific attributes
- Reading instance specific attributes
- Deleting instance specific attributes
- Listing all existing attributes of an instance element (with wildcards)
- Assigning relations
- Deleting relations
- Distinguishing between mandatory and optional relations
- Selection using attribute values

2.7 INHERITANCE SUPPORT

Inheritance relations are currently represented in the model by several application elements with special attributes. As, at the moment, only one application element can be requested at a time and as only values of attributes from this specific application element can be requested, an "object" from an inherited class must be loaded in a procedure that is comprised of several steps.

These circumstances call for more support through the sever in the future. When accessing instances of an inherited class, the server should also be able to return the attributes of the superclass. In addition, when accessing the superclass, it should be possible to obtain all information (attributes) concerning the instances in a subclass.

The inheritance depth is limited to one step, i.e. inheritance is only possible from the superclass, but not from a subclass.

2.7.1 SOFTWARE DESIGN CLARIFICATIONS AND LIMITATIONS

- A subclass can only be inherited from one superclass. No multiple inheritance allowed.
- Superclasses are not abstract classes, they are instantiable.
- For the storage of inherited classes the ATF specification need not be extended.

2.7.2 EFFECTS WHEN USING INHERITANCE

When using inheritance one will observe following effects:

- Application model: The information returned also includes information about inherited application elements.
- Reading access through the RPC-API: The full support can only be provided in connection with AOP_GetValE. Accessing application elements through AOP_GetVal only allows to query information about one application element, i.e. either that portion of the superclass that is common to all types or only the instances of one type can be queried with one request.
- Writing access through the RPC-API (AOP_PutVal): The application element must be specified in the request. The server distributes the attributes to the relevant physical database tables according to the superclass and subclass definitions.
- Instance attributes: These are stored with the relevant application element.
- References: [1(optional):1] relation between subclass and superclass.

2.7.3 APPLICATION MODEL

Inherited classes are defined as application elements with separate application element IDs. The same base ID is assigned to these application elements (subclass) as to the superclass. The connections between the superclass and subclass (inheritance relation) are determined via the references (see *2.7.7 Relations*). The [1(optional):1] relation is relevant, which points from the ID attribute in the subclass to the superclass.

The list of attributes includes the attributes of the superclass and subclass.

The application element representing the superclass is extended by an attribute that describes the instance type. A base attribute (objecttype) is dedicated for this purpose, so that clients that use the AOP_GetVal function to access application elements (superclass and subclass) know which subclass they should address in order to get all attributes of an instance. The AID of the application element of the class (superclass, subclass) to which the instance belongs is used to identify the type.

Attention: The definition of the inheritance support allows multiple use of base IDs that so far could be used only once in the model (e.g.: BE_MEA – measurement). This is because the inherited subclasses have the same base ID as the superclass.

2.7.4 PHYSICAL STORAGE

A subclass of an application element is implemented in the database using a new table which maintains the additional attributes. The assignment between the superclass and subclass is done via the ID attribute which is also available in the subclass table. For each inherited application element that represents a subclass, a view of the tables of the superclass and subclass is created. These two tables are joined via the ID and the instance type, i.e. the common part of an instance has the same ID in the superclass as the specific part in the inherited application element.

The name of the table implementing the subclass is stored in SVCENT as DBTNAME. The name of the view is derived from the table name, preceded by the prefix "SVC_".

For inherited application elements there are no sequences for ID creation because the ID of the superclass is used instead.

In SVCATTR only the attributes of the subclass are entered (including the ID attribute), i.e. only the attributes of the inherited application element as well as the attributes of the superclass (not including the ID attribute) are added internally. The special handling of the ID attribute is necessary because the ID attribute in the superclass and the subclass can have different implementation names. In addition, the ID attribute of the subclass constitutes a "foreign key" to the superclass (*also see section 2.7.7 Relations*) which is also stored in the meta information.

When defining the application elements (superclass, subclasses) it must be ensured that no naming conflicts occur. The application and implementation names for the application attributes in the database tables must be unique within the superclass and the inherited subclass.

Attributes of the type DT_BLOB are only allowed either in the superclass or the subclass - one instance can always have only one attribute of the type DT_BLOB.

Note: The type DT_BYTESTR must not be used as data type for attributes of an application element.

EXAMPLE: **STORAGE OF INHERITED ELEMENTS**

The application element "Drivevalues" is a specialization of the application element "Capture“. This circumstance should now be implemented by means of the Inheritance Support.
The keywords used (like create, alter,...) are taken from the SQL89 specification (see normative references) and are typically used to manage tables in databases.

- The application element "Capture“ is extended by the attribute "ObjectType“:

```
alter  table CAPTURE add (OBJECTTYPE number(10) default 4);
create index IDX_CAPTURE_OBJECTTYPE on CAPTURE(OBJECTTYPE);
insert into svcattr (aid,attrnr,aaname,baname,adtype,dbcname)
       values (4,4001,'ObjectType','OBJECTTYPE',6,'OBJECTTYPE');
```

- Creation of the table for the attributes of the inherited class. The new table is very similar to the table for the "Drivevalues“; an attribute that holds the "CaptureID“ is used instead of the ID attribute "DrivevaluesID“. Thus, the attribute "DrivevaluesID“ is no longer required:

```
create table CAPTURE_S1 (
  ID_CP1  number(10) not null constraint PK_CAPTURE_S1 primary key,
  TRACENAME     VARCHAR2(30),
  VEHICLETYPE   VARCHAR2(30),
  ENGINETYPE    VARCHAR2(30),
  ENGID_CALC    VARCHAR2(30),
  GEARBOXTYPE   VARCHAR2(30),
  GEARBOX_I     NUMBER,
  TYRETYPE      VARCHAR2(30),
  REMAX_I       NUMBER,
  GEARNR        NUMBER(2),
  SLOPE         NUMBER,
  VEHICLEMASS   NUMBER,
  DESCRIPTION   VARCHAR2(255),
  CREATEDATE    VARCHAR2(20),
  CHANGEDATE    VARCHAR2(20),
  CREATORNAME   VARCHAR2(20),
  foreign key(ID_CP1) references CAPTURE(CAPTUREID) on delete
           cascade);
```

- Inclusion of the new application element in SVCENT:

```
insert into SVCENT (aid,aname,bid,dbtname,security)
            values (401,'Drivevalues',3,'CAPTURE_S1',1);

Excerpt from SVCENT:
      AID  ANAME                  BID DBTNAME
      --------------------------------------------------
       12  Capturegroups          2   CAPGROUP
        4  Capture                3   CAPTURE
      401  Drivevalues            3   CAPTURE_S1
        5  Measquantities         4   MEASQUAN
        7  Quantities             11  QUANT
```

- Inclusion of the subclass attributes in SVCATTR. The ID attribute is taken over from the superclass, the subclass attributes are taken over from the original application element "Drivevalues“ (except for the ID attributes):

```
insert into svcattr(aid,attrnr,aaname,baname,faid,funit,adtype,
       aflen,dbcname)
select 401,attrnr,aaname,baname,4,funit,adtype,aflen,'ID_CP1'
       from svcattr
       where aid=4 and dbcname = 'CAPTUREID';
insert into svcattr(aid,attrnr,aaname,faid,funit,adtype,aflen,dbcname)
select 401,attrnr,aaname,faid,funit,adtype,aflen,dbcname
       from svcattr
       where aid=25 and upper(dbcname) not in('CAPTUREID',
             'DRIVEVALUESID');

Excerpt from SVCATTR:
  AID  ATTRNR AANAME        BANAME FAID FUNIT ADTYPE AFLEN DBCNAME
  --------------------------------------------------------------------
  401  108    CaptureID     ID     4    0     6            ID_CP1
  401  362    DvTraceName               0     1            TraceName
  401  364    DvEngineType              0     1            EngineType
  401  365    DvEngid                   0     1            Engid_Calc
  401  366    DvGearboxType             0     1            GearboxType
  401  367    DvGearbox_I               0     3            Gearbox_I
  401  368    DvTyreType                0     1            TyreType
  401  369    DvRemAx_I                 0     3            RemAx_I
  401  370    DvGearNr                  0     6            GearNr
  401  371    DvSlope                   0     3            Slope
  401  372    DvVehicleMass             0     3            VehicleMass
  401  373    DvDescription             0     1            Description
  401  374    DvVehicleType             0     1            VehicleType
  401  375    DvCreateDate              0     1            CreateDate
  401  376    DvChangeDate              0     1            ChangeDate
  401  377    DvCreatorName             0     1            CreatorName
```

➢ Creation of a view of superclass and subclass:

```
create or replace view SCV_ CAPTURE_S1 as
  select ID_CP1,…. (all attributes of the superclass and subclass –
                  but only one ID attribute)
         CHANGEDATE,
         CREATORNAME
         from CAPTURE,CAPTURE_S1
         where ID_CP1 = CAPTUREID
         and OBJECTTYPE = 401;
```

2.7.5 RESTRICTIONS OF THE RPC-API

READING ACCESS THROUGH AOP_GETVALE

Full inheritance support can only be implemented with help of AOP_GetValE(). This function offers the possibility to return the result in several parts (*result sets*). Result sets are necessary because the list of attributes depends on the instance type. A separate, homogeneous result set is created for each instance type.

The interface handling differs, depending on whether a superclass or subclass is accessed.

Subclass: For access to a subclass, all attributes, both of the superclass and of the subclass, are available. This means that, if the report list is empty (Select *), all attributes for the instances of this subclass are returned as well.

Superclass: When accessing the superclass and subclass(es), the result can consist of several parts. The superclass and subclass(es) are accessed if the request structure of AOP_GetValE() specifies either the application element IDs of the superclass and of one or more subclass or attributes from the superclass and from one or more subclass(es) in the report list. For each class that appears in the result, a separate result set is returned. The instances are combined into homogeneous result sets according to the instance type. If only attributes from the superclass are requested, the result is returned in one single result set.

If only the application element ID of the superclass or attributes of the superclass are specified in the request structure, only information from the superclass will be accessed. However, all instances are visible, also those belonging to a subclass. If the result is to be restricted to the instances of the superclass, a selection (application attribute with basic attribute name OBJECTTYPE = application element ID of the superclass) needs to be specified explicitly.

READING ACCESS THROUGH AOP_GETVAL

When accessing an application element by using AOP_GetVal() that represents a superclass, only the attributes of the relevant application element may be used. For access to a subclass, however, all attributes (from superclass and subclasses) are available. The function can only return information from the application element that has been addressed. When the superclass is accessed, all instances are visible, also those belonging to a subclass. As in the new interface, by explicitly specifying a selection criterion, the view can be limited in such a way that only the instances of the superclass are accessed.

WRITING ACCESS

When writing, the application element ID for the given instance type must be indicated. The server distributes the attributes among the relevant application elements and database tables of the superclass and subclass and enters the proper instance type (class AID) in the type attribute (base attribute name: OBJECTTYPE) of the superclass. The instance's ID is determined by means of the superclass sequence and is used for both the superclass part and the subclass part.

2.7.6 INSTANCE ATTRIBUTES

Instance attributes are stored in the SVCINST table using the key *application element ID and instance ID*, i.e. the AID of the relevant class is stored as key in SVCINST.

2.7.7 RELATIONS

Only relations to a superclass and from a superclass are allowed. The inheritance from a superclass automatically establishes a [1(optional):1] relation between the superclass and subclass (inheritance relation), which is also entered in the meta information. The application element ID of the superclass is entered as "foreign“ application element ID (FAID) of the ID attribute of the subclass. No further relations to a subclass, or from a subclass, are allowed.

2.8 THE USE OF ATF IN THE ODS ARCHITECTURE

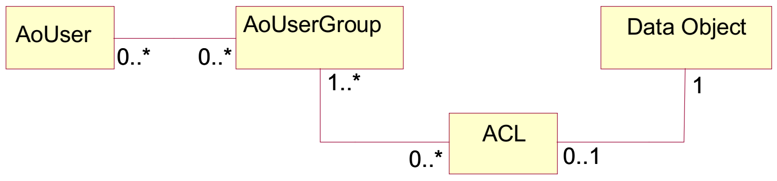
The use of the ASAM Transport Format (ATF) is described in detail in *ASAM ODS Version 5.0, Chapter 5, ATF/CLA* and *ASAM ODS Version 5.0, Chapter 6, ATF/XML.*

2.9 SECURITY CONCEPTS OF ASAM ODS

2.9.1 GENERAL

ODS access control for data objects is included in both APIs (the OO-API and the RPC-API).

2.9.2 BASIC CONCEPT



2.9.3 EXPLANATION OF BASIC TERMS

AoUser:

- An instance of an application element derived from the base element AoUser denotes an individual user as used by a client for identification when accessing data via the ODS interface.

AoUserGroup:

- An instance of an application element derived from the base element AoUserGroup denotes a user group that share the same access rights..
- A user may belong to one or more than one user group.
- A user's right to access data objects is defined by his membership to user groups.
- A user group is an abstract term, which, from its concept, does not refer to particular organizational units.
- This concept does not provide any further structuring of user groups (i.e. no sub-groups).

  Notes:

  - Depending on company conventions, a user group may assume different meanings, for example: “Project”, “Department”, “Role”.
  - Depending on company conventions, organizational structures can be reflected by using naming conventions for user groups, for example: “DepartmentX_Engineer”, “ProjectY_Operator”.

Data Object:

- The access rights of user groups for a data object are defined using an Access Control List (ACL).

Access Control List (ACL):

- The Access Control List determines the access rights that particular user groups are granted for a data object.

2.9.4 ACCESS CONTROL FOR DATA OBJECTS

Access to data objects is controlled as follows:

- **Instances and attributes (except references) are protected via an ACL.**
- **Attributes are protected for a whole application element, not for a single instance.**

In case of instances which are not protected via an ACL but by a parent-child-relationship, the instance inherits the access protection from the parent instance

Notes:

- Only instances are protected via ACL but not the references to instances.
- Application attributes of single instances are not protected individually.
- Instance attributes are not protected.
- Application elements (e.g. adding new attributes) may only be changed by superusers.
- Only a superuser may configure how ACLs are used to protect the data.

EXAMPLE:

Let the instance "MyMeasurement" be child of "MyTest".

If the instance “MyMeasurement” does not have an ACL of its own but is defined via the application model as child of “MyTest”, the access protection for “MyTest” will be assumed implicitly.

An application attribute like “measurement_begin” of the instance “MyMeasurement” cannot be protected individually. Either they are protected for all instances or for no instance.

The instance attribute “MyAttribute” of the instance “MyMeasurement” is not protected at all.

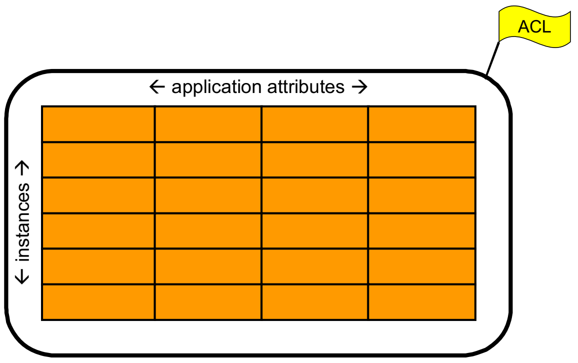
ACCESS PROTECTION VIA ACL

The three data objects described in the following can be protected via an ACL. It must be configured which of these data objects should actually be protected in a system.

ACL Protection on an Application Element (Protection of all of its instances)

In this case all instances of that application element will be protected by the ASAM ODS server according to the access rights specified in the corresponding ACL. In case the physical storage is implemented according to chapter 03 (Physical Storage) all instances of an application element are stored in one table (one row for each instance, application attributes being placed in one column each) and protection is set to the complete table of that application element (marked orange in the figure).

This type of access protection will be referenced by "element security" throughout this chapter.



ACL on application element:
→ protect all entries in a table

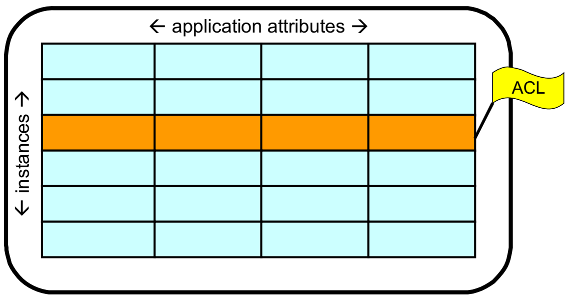
EXAMPLE:

The ACL entry “Test: GroupX (Read)” defines read access to <u>all</u> tests for this user group.

ACL Protection on an Instance Element (individual protection on an instance)

In this case only one instance of an application element will be protected by the ASAM ODS server according to the access rights specified in the corresponding ACL. In case the physical storage is implemented according to chapter 03 (Physical Storage) all instances of an application element are stored in one table (one row for each instance, application attributes being placed in one column each) and protection is set to one row in the table of the application element (marked orange in the figure).

This type of access protection will be referenced by "instance security" throughout this chapter.



ACL on instance
→ protect a single entry (row) in a table

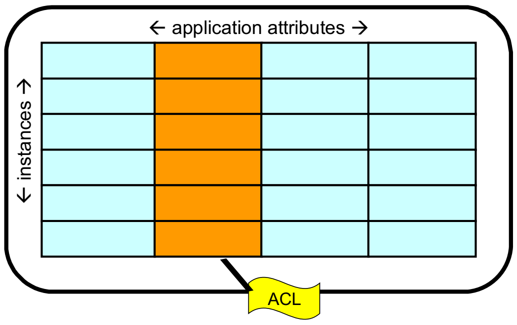
EXAMPLE:

The ACL entry "MyTest: GroupX (Read)" defines read access to <u>exactly this test</u> for this user group.

ACL Protection on an Application Attribute

In this case only one application attribute of an application element will be protected by the ASAM ODS server according to the access rights specified in the corresponding ACL. Protection however will cover the application attribute for all instances of that application element. In case the physical storage is implemented according to chapter 03 (Physical Storage) all instances of an application element are stored in one table (one row for each instance, application attributes being placed in one column each) and protection is set to one column in the table of the application element (marked orange in the figure).

This type of access protection will be referenced by "element security" throughout this chapter.



ACL on application attribute:
→ Protect a column for all entries in a table

EXAMPLE:

The ACL entry "measurement_begin: GroupX (Read)" defines read access to the attribute "measurement_begin" in all measurements for this user group.

Hint for implementation:

The ACL on an application element (access protection for all of its instances) is mainly required to control the insert right for data that is not structured hierarchically. For the sake of clarity, it is recommended not to configure “ACL on application element” and “ACL on its instances” at the same time in a data model.

CONFIGURATION OF ACL ACCESS PROTECTION

Simple configuration keeps administration easy whereas selective (complex) configuration meets selective requirements.

ACL access protection for a particular element and its attributes is configured using the SVCENT table (see 0):

For each application element:

- Are all associated instances protected (via ACL on application element) ?
- Is each individual instance of the application element protected (via ACL on the instance element itself) ?

For each application attribute:

- Is this attribute protected in all instances (via ACL on application attribute) ?

If access protection has been enabled for instances of an element, all instances of this element must have an ACL.

If access protection has been enabled for attributes of an element, all attributes of this element must have an ACL.

Example of simple access protection configuration:

- **Test series**: Instance protection via ACL (test series can be protected individually).
- **Test**: Instance protection via ACL (tests can be protected individually).
- **Measurements and subordinate measurement data**: No protection (access protection inherited from tests).
- **Attributes**: No protection (no protection on single attributes).
- **Other**: Application element protection via ACL (e.g. general protection on quantities).

2.9.5 BASIC RIGHTS

Five basic rights are distinguished:

Read	Update	Insert	Delete	Grant

Read:

- Read-right on application element: The client may read all instances of this application element.
- Read-right on instance: The client may read this instance.
- Read-right on attribute: The client may read this attribute of all instances of the application element.

Update

- Update-right on application element: The client may modify all instances of this application element.
- Update-right on instance: The client may modify this instance.
- Update-right on attribute: The client may modify this attribute of all instances of the application element.

Insert:

- Insert-right on application element: The client may create new instances of this application element.
- Insert-right on instance: The client may create child instances for this (parent) instance.
- Insert-right on attribute: This does not make sense and will be ignored by the server (should not be supported by administration tool).

  Notes:

  - Creating a child-instance means creating a new instance or reassigning an existing child to a different parent.
  - An Insert-right on an instance for an application element which is not a parent in any parent-child relationship does not make sense and will be ignored by the server (should not be supported by administration tool)

Delete:

- Delete-right on application element: The client may delete any instance of this application element.
- Delete-right on instance: The client may delete the specific instance.
- Delete-right on attribute: This does not make sense and will be ignored by the server (should not be supported by administration tool).

Grant

- Grant-right on application element, instance or attribute: Access rights may be passed on. This right is interpreted by the server as follows: If a user group has GRANT right on a data object, any of its members may pass the group’s rights on the data object to other user groups.

The following table shows the possible combinations of the access rights:

Read	Update	Insert	Delete	Meaning:
0	0	0	0	No access
0	0	1	0	(Invalid combination)
0	0	0	1	(Invalid combination)
0	0	1	1	(Invalid combination)
0	1	0	0	(Invalid combination)
0	1	1	0	(Invalid combination)
0	1	0	1	(Invalid combination)
0	1	1	1	(Invalid combination)
1	0	0	0	Read
1	0	1	0	Read and Insert (perhaps for test bed operator)
1	0	0	1	Read and Delete (e.g. administrator)
1	0	1	1	Read, Delete and Insert (e.g. administrator)
1	1	0	0	Read and Update (e.g. engineer)
1	1	1	0	All except Delete (e.g. engineer)
1	1	0	1	All except Insert
1	1	1	1	All access rights

To avoid “blind access”, all combinations of Update, Insert and Delete rights without granting the Read right are considered to be invalid (Configuration tools should not accept such combinations). In case of such combinations, the server will deny access and return an error message.

Note:

Every desired right needs to be entered separately. There is not a hierarchy of rights in the sense of “Right A automatically implies Right B”.

Hint for implementation:

To avoid complex and unnecessary attribute protection, access to some particular attributes (IDs, references) should be denied by client application programs rather than via attribute protection. Client Log-on to the Server

When a session is opened, the client logs on by indicating the userID and password. The server checks if the information is valid (please see section 2.9.8, “Authentication”). If the information is invalid, an error message will be returned.

For client requests during the same open session, it is no longer necessary for the client to provide userID and password.

2.9.6 CHECKING THE ACCESS RIGHTS FOR A DATA OBJECT

When a client accesses an instance element/attribute, the server performs the checks described below. What will actually be checked depends on the configuration in the SVCENT table. For the details of these checks, please also see section 2.9.5, “Basic Rights”.

The desired access is only permitted if all of the following checks are ok (restrictive approach!), otherwise the server will return an error message.

1. **ACL on application element:** Is it permitted to access any instances of this application element?

2. **ACL on application attribute:** Is it permitted to access this attribute of the application element?

3. **ACL on instance element:** Is it permitted to access this instance element (in case of Insert: this parent instance element)?

   or (if the instance element has no ACL of its own but is the child of a parent instance):

4. **ACL on parent instance element:** Is it permitted to access the parent instance element (This check may be repeated over several levels)?

In case of elements which do not have element security or instance security of their own and for which a parent-child relationship is defined by the base model, the server will search the hierarchy until an element with element security or instance security is found or until the highest element in the hierarchy is reached. If no element is found even in the highest element, access will be granted for all elements in the hierarchy.

Note: When accessing measurement data (SVCVAL) the access control on MEQ will be used.

2.9.7 CREATING A NEW DATA OBJECT

The following rules are valid for elements/attributes which, according to configuration in SVCENT, need to have an ACL of their own.

To create a new instance element, the access rights for the new object need to be defined. In this connection, different requirements on the part of the various vehicle manufacturers must be met. The rules described below are to fulfill all of these requirements.

Hint for implementation:

To avoid complex configuration, a system will preferably be configured in such a way that only one of these rules is applied to each element.

To determine the access rights for newly created data, the server applies the rules described below (in the indicated sequence).

EXPLICIT SPECIFICATION OF THE NEW ACL BY THE WRITING CLIENT

If the writing client explicitly specifies the desired ACL prior to the insertion, the server will in any case use the indicated ACL.

IMPLICIT RULES FOR THE NEW ACLS

The server applies the following rules in the indicated sequence:

If the criteria below are met, …	… the access rights will be as follows:
1. The application element of the new instance has a configured “ACL template” reference to an instance of some application element.	The ACL template of the referenced instance will be copied as the ACL of the new instance.
2. An ACL template exists for the application element.	The ACL template of the application element will be copied as the ACL of the new instance.
3. This instance has a parent instance, and this parent instance has an ACL.	The ACL of the parent will be copied as the ACL of the new (child) instance.

EXAMPLE: **FOR CASE 1**

For “Measurement”, we want to define an appropriate reference to “Test”: For a new measurement “MyMeasurement”, the system will search this reference for an appropriate test “MyTest” and will then use the ACL template attached to this test as the ACL for “MyMeasurement”.

EXAMPLE: **FOR CASE 1**

For “Test”, we want to define an appropriate reference to "TestOrder": For a new test “MyTest”, the system will search this reference for an appropriate test order “MyTestOrder” and will then use the ACL template attached to this test order as the ACL for “MyTest”.

For application elements which are not organized hierarchically in parent-child relationships, it makes sense to attach an ACL template to the application element and thus use case 2.

EXAMPLE: **FOR CASE 2**

All quantities may be created using the same ACL. It is possible to modify access protection (e.g. define tighter protection) for individual quantities at a later time.

Case 3 is modeled according to the Windows NT® file system handling, in which the ACL of a parent instance (NT: of a directory) is copied as the ACL of the child instance (NT: of the file in the directory).

EXAMPLE:	FOR CASE 3
	An administrator creates new test series and adjusts their ACLs to the desired access rights: all new subordinate tests will be created automatically with the same access rights.

PROCEDURE IF NONE OF THE SPECIFIED RULES APPLIES

If new instances are to be created (for which instance protection has been defined in SVCENT) and if no ACL template is available (error during system setup!), the instance will be created with an empty ACL (nobody has access) and an error message will be output to the client. The superuser can modify the ACL later.

An empty ACL contains no entries for the specified data object.

Note for server implementation:

If the rights for some data object are set to 0 (no access) for some particular user group the server should remove this ACL entry from the ACL, because a missing ACL entry has the same meaning as “no access “.

2.9.8 OTHER BASIC RULES

RULES FOR IMPLEMENTING ODS SECURITY

The following rules ***must*** be implemented by the ODS server to support ODS security:

1. The attribute password of AoUser must not be transmitted through the API. It must not be visible in the model information.

2. Regardless of the security settings (READ bit) the server must return the following base elements:
   - AoUser
   - AoUserGroup
   - AoEnvironment
   - AoTest
   - AoSubTest
   - AoMeasurement
   - AoMeasurementQuantity
   - AoQuantity
   - AoUnit
   - AoPhysicalDimension

3. Regardless of the security settings (READ bit) the server must return the following base attributes:

- ID
- Name
- Version
- All basic reference attributes
- Objecttype

The above mentioned rules 2.) and 3.) apply only for the READ bit of the ACL. The server must return these elements and attributes even when the READ bit is NOT set. On the other side, the server must check all other security bits (UPDATE, DELETE, INSERT, GRANT) properly. Security Manager Clients can use this rules and should not allow the READ bit to be removed from the above mentioned elements/attributes.

AUTHENTICATION

The server performs authentication via userID and password.

- A server must be able to read a password from a database (password attribute of AoUser).
- If the password comparison with the password from the database fails, the server will refuse to open a client session.
- Passwords must be stored encrypted except when they are initialized for the first time. With the first password-change the changed password must be stored encrypted. An ASAM ODS server must behave as follows: First he must check against the plain password. If this check fails he must check against the encrypted password.

A column in the corresponding database table is named "Password". The encrypted password is stored. In addition, the column is locked for normal ODS access.

The password cannot be read via ODS but can only be set (using an appropriate method).

Note: Currently no external authentication is supported.

OWNER

The Owner of a data object is not part of the ODS access control concept. The Owner can be defined as an application attribute for information purposes.

The server sets the value of this attribute at the time when a new instance is created. For clients the attribute is read-only: this is automatically handled by the server.

AUTOMATIC PROGRAMS

For ODS clients working in batch mode in the background, the regular access rules apply: The client needs to log on to the relevant server by indicating the userID and password and is then given the relevant data access rights.

The rules for ACLs of data objects that are newly to be created also apply to these clients.

Hint for implementation:

As, in automatic operation, data objects are usually created for someone, not by someone, automatic clients will be given rather privileged UserIDs for log-on at starting. It is up to client implementation and system configuration to provide the client with the required rights once it has been started.

SUPERUSER

The “Superuser” user group has automatically full access rights to all data.

Only members of this user group are allowed to add further users to the "Superuser" user group.

The “Superuser” user group is identified by a “superuser_flag” base attribute in the AoUserGroup application element. Exactly one user group may have this flag. This must be configured (outside ODS) at system setup, and it is not accessible via the ODS interface.

At least one user must be allocated to this user group. This must be configured (outside ODS) at system setup. The server does not allow the removal of the last existing user in this group.

INSTALLATION OF ACCESS CONTROL

In a system without access control, it must be possible to set up the data access control information via ATF file and then activate it.

SYSTEMS WITH/WITHOUT ACCESS CONTROL

If a server recognizes a system for which access control is defined, it will apply the relevant rules for access protection within this system.

Systems can be set up with or without access control.

2.9.9 PHYSICAL DATA MODEL

The physical data model is described in detail in *Chapter 3 - Physical Storage*.

The impact of security on the ASAM Transport Format (ATF) is described in *Chapter 5 - ASAM Transport Format Classic (ATF/CLA)* and *Chapter 6 - ASAM Transport Format XML (ATF/XML)*.

2.9.10 ACL

An ACL entry creates a reference between a group and a data object and describes the associated rights via a bit mask. All of these entries for a data object form the ACL of this data object. There are the following ACL tables:

- Table SVCACLI for instance protection
- Table SVCACLA for attribute and element protection

SVCACLI:

Logical:	Group ID	Appl. Element ID	Instance ID	Rights
Database-column:	USERGROUPID	AID	IID	RIGHTS

SVCACLA:

Logical:	Group ID	Appl. Element ID	Attr. Name	Type	Rights
Database-column:	USERGROUPID	AID	AANAME	TYPE	RIGHTS

Rights:

In this field (Type = short integer), the five basic rights Read, Update, Insert, Delete, Grant are entered as bits. “Bit set” means “right granted”. Sequence of the rights:

Bit 0	Bit 1	Bit 2	Bit 3	Bit 4
Read	Update	Insert	Delete	Grant

2.9.11 CONFIGURATION OF ACL PROTECTION

ACL access protection for a particular element and its attributes is enabled/disabled in the new integer-type column **SECURITY** in the **SVCENT** table via bit code (bit 0 = off, 1 = on).

Bit #	**Decimal**	**Configuration of:**
0	1	ACL on application element (= protection on all instances)
1	2	ACL on the element's instances (= protection individually on each instance)
2	4	ACL on the element's attributes (= protection on each attribute)

2.9.12 ACL TEMPLATE

An ACL template is a model of an ACL, which describes the rights of a user group to access a (newly to be created) data object.

ACL templates are stored in the same format as ACLs, with the difference that, for the ACL template, the indicated instance element is the one at which the ACL template is stored whereas for the ACL, this is the one that is to be protected by the ACL.

ACL templates are stored as follows:

In the table **SVCTPLI** for ACL templates, which are attached to instances or application elements.

SVCTPLI:

Logical:	Group ID	Appl. Element ID	Instance ID	Ref. Appl-ID	Rights
Database-column:	USERGROUPID	AID	IID	REFAID	RIGHTS

If an entry in **SVCTPLI** is attached to an application element only, the InstanceID must be set to 0.

In the **SVCATTR** table a new column **ACLREF** with data type short-integer (values: 0/1) must be added. This flag tells the server which reference-attribute(s) must be resolved to find the ACL-Template that will be used while creating a new instance.

Note: The server will set REFAID to the application element id that refers to AID/IID, to distinguish between different ACL-Templates for different (more than one) referencing elements.

2.9.13 USERS AND USER GROUPS

For checking username and password a mandatory base attribute “password” is included in AoUser.

Both, username and password are case sensitive.

Exactly one user group must be the superuser-group. To identify the superuser-group a mandatory base attribute “superuser_flag“ is included in AoUserGroup. The “superuser_flag” is read-only (a client cannot set/change the value of “superuser_flag”).

2.9.14 ATF FORMAT

Optionally, access protection information can be saved in the ATF file in such a way that it can be restored completely in the database during later import. It should be noted that the access control information will appear as plain ASCII text within the ATF file.

Depending on the purpose of the ATF export two cases can be distinguished:

- Export for data exchange with other systems, companies, etc.:
  Saving of the data access control information in ATF is not required.
- Export for archiving and later re-import into in the same system:
  The data access control information is saved in the ATF file (-the information must be saved in such a way that it can be interpreted completely during a later import).

This also has influence on the import of ATF files:

- Import of exported files (without data access control information):
  For the creation of access rights the same rules apply as for the creation of new data.
- Import of archived files (including data access control information):
  During ATF file import it is necessary to take over the ACL information for data objects from the ATF file and enter it into the database. If groups do not/no longer exist that have been referenced in the ATF file, the ACL entries will be skipped. However, the data objects themselves must be imported in any case (if necessary, with an empty ACL).

The following (if any) is stored in the ATF file:

- ACL of instances
- ACL templates of instances

For the export/import of data from/into a database into/from the ATF file, the user is required to have the necessary rights to access data objects in the database (for access to the ATF file, no right is required). In general, the user needs to have the following access rights:

- for export: Read,
- to delete exported data objects: Delete,
- for import: Insert and Update.

Notes:

- ACLs and initial rights lists of application elements or application attributes are not stored in ATF because such information always refers to all instances of an application element whereas ATF export concerns only one or several individual instances.
- For data exchange between different systems it makes sense not to store the access protection information and, for import into the database, to set up application dependent rules for access to the imported data.
- For the archiving and later re-import of the data in the same system it makes sense to store the information in such a way that it will be completely available after import at a later time.
- For export, the ATF file is not encrypted.

For a description of the format for the data access control information in ATF, see the document *Chapter 5 - ASAM Transport Format Classic (ATF/CLA)* or the document *Chapter 6 - ASAM Transport Format XML (ATF/XML)*.

2.10 GLOSSARY

Activity of Access Control	The activity of the access control describes the handling of access information stored in ASAM ODS by the →API.
API	The application programming interface (API) provides methods for the application to create, store and read the application model and methods to store and read data stored with this application model.
Application element	Element of the application model which defines the application-specific attributes in addition to the base attributes inherited from the →base element.
Application model	The application model is an application-specific data model which is created from the →base model.
ASAM ODS OO-API	The ASAM ODS OO-API defines platform-independent accesses to the ODS data storage using an object-oriented approach.
ASAM ODS-Data Server	Standard →data server of ASAM ODS for access to the storage formats defined by ASAM ODS (database, file, ...)
ASAM ODS-interpretable Data Format	An (existing) data format which is different from ASAM ODS will be called ASAM ODS-interpretable if it is possible to access the format by adding →meta-information to an →ASAM ODS-server without the need to create an own foreign format server.
Attributes	The term attributes combines all information which arises during a measurement and which does not belong to measured data, which means it is not stored in the →value matrix. This kind of data usually contains only single values per measurement and not a value vector. The attributes are stored within →instance elements.
Base Element	Element of the →base model; it specifies the generally available attributes of all derived →application elements.
Base Model	The base model is a general, application-independent, logical data model which defines the structure of the data to be processed by ASAM ODS.
Data Server	The data server manages the actual accesses to the corresponding data formats.
Environment	An environment consists of a specific →application model and all data stored in it.
Foreign Formats	(Existing) data formats are called foreign formats if they can be accessed through the ASAM ODS APIs using a special →data server.
Instance Element	A data set of an →application element.

Local Column	A local column corresponds to a column in the →value matrix; a column will be identified by the local column number.
Measured Data	All data which is stored in the →value matrix is called measured data, this means all data of the →measurement quantities. Usually, these are complete data vectors. They include the actual measurement results and reference quantities and also the processed data (which are results of mathematical calculations). All measured data of a measurement will be combined in the value matrix.
Value Matrix	Virtual matrix (not physically existing in the ODS server) which combines all →measured data that belong to a measurement.
Measurement Quantity	Base element which contains the information of a measured quantity; the data of a measurement quantity form a column of the →value matrix.
Meta Information	Standardized description of an (existing) data format; using this, an →ASAM ODS-data server is able to access data of such a format.
Submatrix	Submatrices are usually homogeneous parts of the →value matrix, which means sections without gaps (undefined values); only the submatrices will be stored physically.
Submatrix Links	Rules for the set-up of the →value matrix from the single →submatrices.
Value Matrix	The value matrix is a data structure which is used for the exchange of →measured data between the application and the →API. The value matrix provides logically a matrix but is able to contain different data types at the same time. The value matrix contains, besides the pure data including the value flags also minimal information of the →measurement quantities (quantity ID, value type and data type), to enable access.

2.11 REVISION HISTORY

Date Editor	Changes
2003-06 P. Voltmann	New base elements AoLog, AoParameter, AoParameterSet, AoNameMap, AoAttributeMap and AoExternalComponent added Description of Data Type Conversion added Tables for Base Elements completed and extended Description of ASAM Data Type harmonization added
2003-10-16 R. Bartz	Several errors have been fixed and explanations have been changed to make things clearer Duplicate sections have been deleted
2003-11-22 R. Bartz	Completely restructured and extended the information on data types
2003-12 R. Bartz	The section on the APIs and the section on Inheritance has been restructured and revised The section on parameters has been extended The glossary and some information on ‘old’ vs. ‘new’ APIs has been revised Several missing base attributes have been added to the base elements descriptions
2003-12-30 R. Bartz	The **Release** version has been created
2004-05 R. Bartz	The detailed description of Base Elements has been moved to Chapter 4 The information on Security (former Chapter 7) has been introduced as section 2.9 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 3
PHYSICAL STORAGE

Version 1.1

Association for **S**tandardisation of
Automation and **M**easuring Systems

Dated: 30.09.2004

© ASAM e.V.

Status of Document

Reference	ASAM ODS Version 5.0 Physical Storage
Date:	30.09.2004
Authors:	Horst Fiedler, TIFFF; Gerald Sammer, AVL; Karst Schaap, HighQSoft
Type:	Specification
Doc-ID:	ASAM_ODS 50_CH03_Physical_Storage.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 (odsftr@asam.net) to make sure this issue will be addressed within the next review cycle.

Contents

3 PHYSICAL STORAGE 3-3
3.1 INTRODUCTION 3-3
3.1.1 GENERAL 3-3
3.1.2 DATA TYPES USED 3-4
3.2 DESCRIPTION OF SVC TABLES 3-5
3.2.1 SVCENT 3-5
3.2.2 SVCATTR 3-7
3.2.3 SVCREF 3-12
3.2.4 SVCVAL 3-14
3.2.5 SVCINST 3-17
3.2.6 SVCENUM 3-18
3.2.7 SVCACLI 3-25
3.2.8 SVCACLA 3-26
3.2.9 SVCTPLI 3-27
3.3 STORAGE OF THE ATTRIBUTE VALUES 3-28
3.3.1 STORAGE OF SINGLE VALUES 3-28
3.3.2 STORAGE OF BYTESTREAM VALUES 3-28
3.3.3 STORAGE OF BLOB VALUES 3-28
3.3.4 STORAGE OF ARRAY VALUES AND OBJECT VALUES FOR ATTRIBUTES 3-28
3.3.5 STORAGE OF ATTRIBUTE VALUE FLAGS 3-30
3.3.6 STORAGE OF RELATIONS TO THE SUPERCLASS 3-31
3.4 THE MIXED-MODE-SERVER 3-32
3.4.1 MODEL OF MIXED-MODE-SERVER 3-32
3.4.2 USE CASES 3-33
3.4.3 APPROACH FOR SPECIFYING MIXED-MODE-SERVER 3-35
3.4.4 SEGMENTATION WITHIN THE MIXED-MODE-SERVER 3-37
3.4.5 USE-CASES 3-39
3.4.6 CONSIDERATIONS FOR MIXED-MODE SERVER IMPLEMENTATIONS 3-40
3.5 REVISION HISTORY 3-41

Scope

This document describes the Physical Storage of the ASAM ODS Version 5.0 including some examples and the impact on the APIs (especially the RPC-API).

Intended Audience

This document is intended for implementers of ASAM ODS Version 5.0 server. It shall be used as a technical reference with examples how to store the required additional information used in 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

3 PHYSICAL STORAGE

3.1 INTRODUCTION

3.1.1 GENERAL

This chapter describes how measurement data and results are (physically) stored in a relational database to comply with the ASAM ODS standard.

For the physical storage of ASAM ODS information nine tables are used in addition to those that hold the result data.

These tables are named SVCxxx, where xxx describes the kind of table.

The nine tables are:

- **SVCENT** for storage of application elements (metamodel)
- **SVCATTR** for storage of application attributes (metamodel)
- **SVCREF** for storage of references (metamodel)
- **SVCVAL** for storage of values (submatrix and local column)
- **SVCINST** for storage of instance attributes
- **SVCENUM** for storage of enumerations
- **SVCACLI** for storage of security data for protection of instances
- **SVCACLA** for storage of security data for protection of elements and attributes
- **SVCTPLI** for storage of ACL templates

The sections describe those tables in detail. The scripts in that section are specified for Oracle database systems version 7.3 and higher. They should serve as examples for other database systems.

A subsequent section describes how the values of application attributes are stored, especially if they cannot be placed immediately into the corresponding tables.

A final section explains the storage mechanism in the so-called "mixed-mode", where mass data are stored externally while most application model information is kept in the relational database.

3.1.2 DATA TYPES USED

ASAM has specified standardized ASAM data types which have been published end of 2002. These data types show standardized names, always beginning with A_. Those data types will be used by ASAM ODS consistently in all new specifications in the future.

The Physical Storage based on relational databases has been developed and specified several years ago. At that time no standardized data types have been available ASAM-wide. Therefore ASAM ODS decided to define data types that cover the needs of ASAM ODS. These data types always start with either T_ (in case they are single valued/structured types) or S_ (in case they are sequences resp. arrays of single valued/structured types).

The Physical Storage described in this chapter has been implemented in ODS servers and installed in a quite large number of companies. Replacing the T_ types by the A_ types would require to modify and exchange those installed servers and the underlying database designs, and also to modify the clients accessing the servers.

That is why ASAM ODS has decided to not introduce the standardized ASAM data types (A_) in the Physical Storage description.

Instead, chapter 2 of the ASAM ODS specification describes the relationship between the T_ types and the A_ types. One should note that the enumeration of the data types and their enumeration names are identical to those specified in the ASAM data type specification, and that the data types themselves are in most cases binary compatible. So a one-to-one mapping is easily possible, if needed.

3.2 DESCRIPTION OF SVC TABLES

3.2.1 SVCENT

This table holds the descriptions of the different application elements. The syntax for creating this table is as follows:

```
create table svcent
(
    AID          integer NOT NULL,    /* Identifier */
    ANAME        char(30) NOT NULL,   /* Unique(!) name of application element */
    BID          integer NOT NULL,    /* ID of base element, only internal) /*
    DBTNAME      char(30) NOT NULL,   /* Name of the table */
    SECURITY     integer,             /* Security mode (bit masked) */
);
```

This specification restricts the length of application elements to a maximum of 30 characters. The respective Base Element IDs (BIDs) are shown in the following table:

BID	Base Element Name
0	AoAny
47	AoAttributeMap
1	AoEnvironment
40	AoExternalComponent
39	AoLocalColumn
43	AoLog
3	AoMeasurement
4	AoMeasurementQuantity
46	AoNameMap
44	AoParameter
45	AoParameterSet
15	AoPhysicalDimension
11	AoQuantity
12	AoQuantityGroup
38	AoSubmatrix
2	AoSubTest
36	AoTest
37	AoTestDevice
23	AoTestEquipment
24	AoTestEquipmentPart
25	AoTestSequence
26	AoTestSequencePart
13	AoUnit
14	AoUnitGroup
21	AoUnitUnderTest
22	AoUnitUnderTestPart
34	AoUser
35	AoUserGroup

The base element ID is to be delivered at the API.

The column *DBTNAME* holds the name of the respective table. For array tables, the naming convention is “<Table name>_ARRAY”. The length of the complete table name **must not** exceed the maximum allowed length within the data base. This means e.g. for Oracle data bases the length of <Table name> must not exceed 24 characters (the maximum table name is 30 characters minus the six characters “_ARRAY”).

The column *SECURITY* holds the access protection for a particular element and its attributes. It is enabled/disabled via a bit code (bit 0 = off, 1 = on).

Bit #	**Decimal**	**Configuration of:**
0	1	ACL on application element (= protection on all instances)
1	2	ACL on the application element's instances (= protection individually on each instance)
2	4	ACL on the application element's attributes (= protection on each attribute)

Example for a SVCENT table:

AID	ANAME	BID	DBTNAME	SECURTITY
1	TestObject	21 (AoUnitUnderTest)	HEADER	0
2	TestRun	36 (AoTest)	STEP_NAMES	0
3	Capture	3 (AoMeasurement)	TEST_STEP	0

3.2.2 SVCATTR

This table holds the descriptions of the different application attributes. The syntax for creating this table is as follows:

```
create table svcattr
(
AID       integer NOT NULL,  /* Identifier of the element               */
ATTRNR    integer NULL,      /* Attribute number for the sequence       */
AANAME    char(30) NOT NULL, /* Unique(!) name of AE attribute          */
BANAME    char(30) NULL,     /* Name of base attribute                  */
FAID      integer NULL       /* Reference to foreign application element */
FUNIT     integer NULL,      /* Reference to unit, if always the same   */
ADTYPE    integer NULL,      /* Data type of attribute, (eg. float, long)*/
AFLEN     integer NULL,      /* Max. length of strings or streams       */
DBCNAME   char(30) NOT NULL, /* Name of column in the respective table  */
ACLREF    integer,           /* attribute for ACL-Template reference    */
INVNAME   char(30) NOT NULL, /* Inverse name of AE attribute            */
FLAG      integer            /* Flags of the attribute.                 */
ENUMNAME  char(30) NULL      /* Name of the enumeration in case ADTYPE is DT_ENUM,
otherwise unused */
);
```

This specification restricts the length of application attributes to a maximum of 30 characters.

FUNIT is the reference to the Id of the unit given in the table of the application element with the base type AoUnit.

In comparison to the ASAM ODS Version 4.0, this table was extended to hold the following additional information: Storage of the inverse relation name for the relations and storage of the application attribute flags, like UNIQUE, OBLIGATORY and AUTOGENERATE.

In the ASAM ODS base model all relations have an inverse relation name. To store the inverse relation name, an additional column had to be defined in the tables *svcattr* and *svcref (see next section)* to store the inverse relation name of a relation. The name of the relation is the name given in the column *aaname* of the table *svcattr* or in the column *refname* in the table *svcref*. Therefore, the tables *svcattr* and *svcref* were extended with a column called *INVNAME* to store the inverse relation name.

The relation attributes of the forward references will have the defined database column name DBCNAME = "NULL". This will indicate that this attribute is not physically available as a column in the table. The same definition will be used for the attribute values, flags and generation_parameters of AoLocalColumn, because these attributes are also not physically available in a table column but stored in the table SVCVAL (or the external component file when using a mixed-mode-server).

In the ASAM ODS base model the base- and application attributes have flags like UNIQUE or OBLIGATORY. These flags must be stored in the table *svcattr*. Therefore, the column *FLAG* was added to the table *svcattr* which contains the information of the application attribute flags.

The following table shows the bits set in this column and their respective meaning:

Bit	Meaning
0	(UNIQUE) This flag tells the server that the values of this attribute have to be unique. 1 means the values must be unique. The server can check this behaviour by database constraints.
1	(OBLIGATORY) This flag tells the server that the values of this attribute are obligatory. 1 means the values must be set. The server can check these behaviour by database constraints.
2	(AUTOGENERATE) This flag tells the server that the values of this attribute has to be generated by the server, e.g. the Id. For each column, there will be a trigger defined within the database or the server.

There is no impact on the ASAM ODS Version 5.0 RPC-API definition. The definitions of the tables for the Version 5.0 RPC-API do not change. All columns of the tables *svcattr* and *svcref* are still available. After the tables are modified, an ASAM ODS ODS-Server Version 5.0 RPC-API is still able to run with the modified physical storage, the new columns of the tables will be ignored.

For details of the flag AUTOGENERATE see the document *ASAM ODS Version 5.0, Chapter 10, The OO-API, section 10.2.5, items ApplicationAttribute_isAutogenerate and ApplicationAttribute_setIsAutogenerate*.

THE DATA TYPES OF THE COLUMN ADTYPE

The column *ADTYPE* specifies the data type with which values of this application attribute will be coded in the database. Instead of specifying the data type as a string it is given as a number (representing its data type enumeration value). The relationship between the enumeration and the data type itself is shown in the following table, where T_xxx are basic data types, S_xxx are sequences of the basic data types, and DT_xxx (resp. DS_xxx) are the names of the data type enumeration:

Name of Data Type Enumeration	Name of Data Type	Description
DT_UNKNOWN (=0)		Unknown data type.
DT_STRING (=1)	T_STRING	String.
DT_SHORT (=2)	T_SHORT	Short value (16 bit).
DT_FLOAT (=3)	T_FLOAT	Float value (32 bit).
DT_BOOLEAN (=4)	T_BOOLEAN	Boolean value.
DT_BYTE (=5)	T_BYTE	Byte value (8 bit).
DT_LONG (=6)	T_LONG	Long value (32 bit).
DT_DOUBLE (=7)	T_DOUBLE	Double precision float value (64 bit).
DT_LONGLONG (=8)	T_LONGLONG	LongLong value (64 bit).
DT_ID (=9)	T_ID	LongLong value (64 bit). Not used. DT_LONGLONG is used instead.
DT_DATE (=10)	T_DATE	Date.
DT_BYTESTR (=11)	T_BYTESTR	Bytestream.
DT_BLOB (=12)	T_BLOB	Blob.
DT_COMPLEX (=13)	T_COMPLEX	Complex value (32 bit each part).
DT_DCOMPLEX (=14)	T_DCOMPLEX	Complex value (64 bit each part).
DS_STRING (=15)	S_STRING	String sequence.
DS_SHORT (=16)	S_SHORT	Short sequence.
DS_FLOAT (=17)	S_FLOAT	Float sequence.
DS_BOOLEAN (=18)	S_BOOLEAN	Boolean sequence.
DS_BYTE (=19)	S_BYTE	Byte sequence.
DS_LONG (=20)	S_LONG	Long sequence.
DS_DOUBLE (=21)	S_DOUBLE	Double sequence.
DS_LONGLONG (=22)	S_LONGLONG	Longlong sequence.
DS_COMPLEX (=23)	S_COMPLEX	Complex sequence.
DS_DCOMPLEX (=24)	S_DCOMPLEX	Double precision complex sequence.
DS_ID (=25)	S_ID	LongLong sequence. Not used. DS_LONGLONG is used instead.
DS_DATE (=26)	S_DATE	Date sequence.
DS_BYTESTR (=27)	S_BYTESTR	Bytestream sequence.
DT_EXTERNALREFERENCE (=28)	T_EXTERNALREFERENCE	External reference.
DS_EXTERNALREFERENCE (=29)	S_EXTERNALREFERENCE	Sequence of external reference.
DT_ENUM (=30)	T_LONG	Enumeration
DS_ENUM (=31)	S_LONG	Sequence of enumerations

This definition of data types is compliant with the one for ASAM ODS 4.1. The data type is also delivered at the RPC interface. The value of the data type is identical to the value of the attribute ‚data type‘ in *AoMeasurementQuantity* and ‚default_data type‘ in ‚*AoQuantity*‘. The values of these constants are described in the document ASAM ODS Version 5.0, *Chapter 10, The OO-API, section 10.3*.

ENUMERATION HANDLING IN SVCATTR

An enumeration is a bundle of fixed name value pairs, an enumeration is used to make a data item readable for humans by using the string representation. The computers will use the value representation because of the more compact storages and the faster comparability.

The column *ADTYPE* of the table SVCATTR may have the value DT_ENUM or DS_ENUM. Therefore, the column *ENUMNAME* was added to the table SVCATTR. This column holds the name of the enumeration. This name matches the name in the column *ENUMNAME* of the new table SVCENUM.

For more information on enumerations in the physical storage see section *3.2.6, SVCENUM.*

Example for a SVCATTR table:

AID	ATTRNR	AANAME	BANAME	FAID	FUNIT	ADTYPE	AFLEN	DBCNAME	ACLREF	INVNAME	FLAG	ENUM NAME
1	1	HeadId	id			6		HEAD_ INDEX			5	
1	2	SerialNo	name			1	32	PART_ID				
3	1	TestId	Id			6		TEST_ INDEX			5	
3	2	TestObj_rel		1		6		HEAD_ INDEX				
1	3	State				1	32	HEAD_ STATUS				

The ID is generated by the server and must be unique, so the flag of the attributes HeadId and TestId are set to 5 (0101).

3.2.3 SVCREF

This table holds the descriptions of the different references (relations) of the application elements. In comparison to ASAM ODS 4.0, this table was extended with the column *INVNAME* to hold the inverse relation name (*see also 3.2.2 SVCATTR*) and the columns *BANAME* and *INVBANAME* to hold the base relations for n:m relations and their inverse.

The name of the base relation for 1:N relations is stored in the column *baName* of the table *svcattr*. As there is no entry in the *svcattr* for N:M relations, the meta information of these relations are stored in the table *svcref*. Therefore, the table *svcref* was extended with the column *baName*.

There is no impact on the ASAM ODS Version 5.0 RPC-API definition. The definitions of the tables for the ASAM ODS Version 5.0 RPC-API does not change. All columns of the tables *svcattr* and *svcref* are still available. After the tables are modified, an ASAM ODS Server based on the Version 5.0 RPC-API is still able to run with the modified physical storage, the new columns of the tables will be ignored.

The syntax for creating this table is as follows:

```
create table svcref
(
AID1      integer  NOT NULL,  /* ID of the first application element      */
AID2      integer  NOT NULL,  /* ID of the first application element      */
REFNAME   char(80) NOT NULL,  /* Name of the reference (relation)         */
DBTNAME   char(30) NOT NULL,  /* Name of the table                        */
INVNAME   char(30) NOT NULL,  /* Inverse name of AE relation              */
BANAME    char(30) NOT NULL,  /* base relation name of AE relation        */
INVBANAME char(30) NOT NULL   /* inverse base relation name.              */
);
```

This table has three columns: two named like the Id in the table *svcattr* and REFNAME. Assuming the table *SVCENT* contains the following entries:

AID	ANAME	BID	DBTNAME	SECURITY
3	Unit	...	...	0
4	UnitGroup	...	...	0
...	...	...	...	0

and SVCATTR contains the following entries:

AID	ATTRNR	AANAME	BANAME	FAID	FUNIT	ADTYPE	AFLEN	DBCNAME	ACLREF	INVNAME	FLAG
3	1	Unitid	id			6		UID			5
3	2	Unit	name			1	32	UNIT			
4	1	UnitGroupId	Id			6		UGID			5
4	2	UnitGroup	name			1	32	UNITGROUP			

and the SVCREF contains the following entries:

AID1	AID2	REFNAME	DBTNAME
3	4	Unit_to_Groups	UNITTOGROUPS
...	...	...	...

then the table UNITTOGROUPS would look like this:

UID	UGID	REFNAME
ID of instance of Unit	ID of instance of UnitGroup	Unit_To_Groups
ID of instance of Unit	ID of instance of UnitGroup	Unit_To_Groups
...	...	---

3.2.4 SVCVAL

This table holds the measured data and the value flags of the data. The syntax for creating this table is as follows:

```
create table svcval
(
    MEQID         integer NOT NULL,   /* ID of Measurement Quantity */
    PMATNUM       integer NOI NULL,   /* ID/number of submatrix */
    SEGNUM        integer NOT NULL,   /* Number of segment */
    VALINDEP      integer NULL,       /* Independent Flag */
    VALEXIMP      integer NULL,       /* Implicit/Explicit Flag /*
    VALBLOBLEN    integer NOT NULL,   /* Length of the BLOB */
    VALBLOB       long raw NOT NULL,  /* Data of the column */
);
```

The name of the local column is identical with the name of the measurement quantity. The data type of the data of the column is given in the attribute derived from the base attribute “data type” of the measurement quantity. The ID/Number (PMATNUM) of the submatrix is given. All local columns with the same length have the same ID/Number of the submatrix. If the same measurement quantity is measured in different rates, there are different entries in the table SVCVAL which differ in the ID/Number of the submatrix.

The number of the segment (SEGNUM) is the number of the data segment. It is not necessary to store all data of one local column into one blob, the data may be split into different segments. The segment number starts with one (1).

The two flags describe the kind of the local column:

VALINDEP	Flag for independent column
VALEXIMP	Flag for implicit column

The independent column flag (VALINDEP) marks columns within submatrices which are regarded to be independent (Set values, scales). .

The implicit column flag changes the interpretation of the values itself. Usually values are stored explicitly. For some kind of data (e.g. timestamps on time driven sampling) the value for each row may be calculated. Following variants are provided (n = row number of submatrix):

value count	value in row n	meaning
1	xn = x1	constant
2	xn = x1 + (n -1) * x2	linear expression
3	xn = x1 + ((n - 1) mod (x3 - x1)/x2) * x2	saw curve

Only numerical types are allowed for implicit storage. The calculation is done using the type defined for the column. The expression (x3 - x1)/x2 is truncated to integer to start each saw curve cycle at x1.

The multidimensional case where multiple scalings are required is not yet defined in this version of ASAM ODS.

VALBLOBLEN gives the number of data items in the field VALBLOB. VALBLOB has the following structure:

Length	Values	Possible Gap	Flags

In the blob only values and flags are stored. The delivered length depends on the type of access (Oracle Inline Code or ODBC).

The Endian Order is identical to the Endian Order of the server.

The length is a four (4) bytes field which gives the real number of bytes in the blob. VALBLOBLEN gives only the number of data items.

EXAMPLE:

VALBLOBLEN = 2500
Data type = DT_FLOAT (4 byte)

→ Length = 2500 * 4 = 10.000

<u>The maximum length of a segment is always limited to 10.000.</u>

This length can be also used for data items of strings. Strings are stored as sequence with the delimiter "\0". Example: `Hallo\0Peter\0Test\0` gives a length of 17, VALBLOBLEN is 3.

If the length is greater than the number of bytes required for the data item, flags are stored in VALBLOB. The number of bytes for flags is the difference between the length and the number of bytes required for the data items. One flag is two (2) bytes or one (1) short. The flags are right aligned in VALBLOB. If the length of VALBLOB is greater than the number of bytes required for the data items and the flags, there will be a gap between the values and the flags. The order of the data items and the flags is left aligned.

EXAMPLE: 2.500 FLOAT VALUES WITH FLAGS

1. Blob:

1666 Values + 1666 Flags = 6664 Bytes for Values + 3332 Bytes for Flags.

If the blob length is set to 10.000, the gap will be 4 Bytes.

If the blob length is set to the actual length of 9996, there will be no gap.

2. Blob:

834 Values (2500-1666).

Blob length = 5004 results in no gap.

If Blob length is from 5006 to 10.000, there will be a gap.

An ASAM ODS server must always be able to read blobs of this kind, but it is up to the server manufacturer whether the server can write blobs of this kind.

HANDLING OF GENERATION_PARAMETERS IN VALBLOB AND VALBLOBLEN

In ASAM ODS Version 5.0 the data are stored with

a) generation_parameters for implicit data

b) the value_sequence for explicit data

c) both for raw data

On the API there shall be no difference. Generation_parameters is a legal attribute for which the client can ask; the server must do the conversion between the physical data and the API.

There is a need to store raw data on external components. Therefore, the sequence representation enumeration must be extended with raw_linear+external and raw_polynomial+external.

In case of raw_polynomial the order of the polynomial is stored in the first parameter and the coefficient-0 is stored as the second parameter (and the coefficient-n as the n+2nd parameter). For formulas, the data will contain a string.

For ASAM ODS Version 5.0 with the RPC-API:

Implicit definition in SVCVAL

Constant:	IMPLFLAG = 1 and one value in value-sequence
Linear:	IMPLFLAG = 1 and two values in value-sequence
Raw:	IMPLFLAG = 1 and three values in value-sequence

For ODS Version 5.0 with the OO-API:

impl	Gen.par 1	Gen.par 2	Gen.par 3			
expl	Value1	Value 2	Value 3	...		
raw	Gen. Par 1	Gen. Par 2	Gen. Par 3	Value1	Value2	…
formula	Formula (string)					

Definition of the parameters:

	Impl. Linear	Impl. Const	Raw linear	Raw polynomial
1	Start value	Const value	Offset	Order
2	Increment		factor	Coeff0
3				Coeff1
4				Coeff2
5				Coeff3
				…

3.2.5 SVCINST

This table holds the instance attributes. The syntax for creating this table is as follows:

```
create table svcinst
(
  AID            integer NOT NULL,   /* Application element id */
  IID            integer NOT NULL,   /* Application instance id */
  NAME           char(30) NOT NULL,  /* Instance attribute name */
  AODT           integer,            /* AODS Data type */
  UNITID         integer,            /* Unit-ID */
  NUMVAL         number(38,10)       /* Numeric with maximum precision */
  TXTVAL         varchar2(255)       /* String value */
);
```

The table looks like this:

Logical:	Appl. Element ID	Appl. Instance ID	Inst. Attr, Name	Data type	Unit ID	Numeric Value	Text Value
Database-column:	AID	IID	NAME	AODT	UNITID	NUMVAL	TXTVAL

UNITID is the reference to the Id of the unit given in the table of the application element with the base type AoUnit.

3.2.6 SVCENUM

This table holds the enumeration data. The syntax for creating this table is as follows:

```
create table svcenum
(
ENUMID           integer NOT NULL,   /* Id of the enumeration. */
ENUMNAME         char(30) NOT NULL,  /* Name of the enumeration. */
ITEM             integer NOT NULL,   /* Value of the item. */
ITEMNAME         char(128) NOT NULL  /* Name of the item. */
);

COMMENT ON TABLE svcenum IS 'ASAM ODS enumeration definitions';
```

This specification restricts the length of enumeration names to a maximum of 30 characters and enumeration items to a maximum of 128 characters.

The new table ***SVCENUM*** has been added to the definition of the physical storage.

DEFINITION OF ASAM ODS ENUMERATION

In ASAM ODS there are several enumerations defined, such as data type enum or seq_rep_enum. In the ATF specification there is a definition how to handle the enumeration, but there was no corresponding definition in the physical storage of relational databases or the API. These enumerations are all ASAM ODS defined enumerations.

An enumeration is a bundle of fixed name value pairs; it is used to make a data item readable for humans by using the string representation. The computers will use the value representation because of the more compact storages and the faster comparability.

Up to now in ASAM ODS the enumerations were represented by a T_LONG value, except in ATF, where they are stored as T_STRING.

The ASAM ODS server needs a definition how to store the bundle of name value pairs. There is a fixed structure for these name value pairs, so a new SVC-Table had to be defined, named **SVCENUM**. This table has four fixed defined columns:

- **ENUMID**, ID of the enumeration.
- **ENUMNAME**, the name of the enumeration.
- **ITEM**, the value of the item.
- **ITEMNAME**, the name of the item.

The column *ADTYPE* of the table SVCATTR may have the value DT_ENUM or DS_ENUM. Therefore, the column *ENUMNAME* was added to the table *SVCATTR*. This column holds the name of the enumeration. This name matches with the name in the column *ENUMNAME* of the new table *SVCENUM*.

Each enumeration must have a name, so people know the meaning; this name is stored in the column *ENUMNAME* of the new table *SVCENUM*.

As already mentioned, an enumeration is a fixed name value pair, therefore, the name is stored in the column *ITEMNAME* and the value will be stored in the column *ITEM*.

As soon as an ASAM ODS server finds a value DT_ENUM or DS_ENUM in the column *ADTYPE* of the table *SVCATTR*, the server will give the data type DT_ENUM of the attribute to the client. A new method *getEnumerationDefinition* at the interface *ApplicationAttribute* will give an object to the client via the new interface *EnumerationDefinition*. This new interface allows the client to access the name value pairs.

There are also methods to get the list of all possible names of the items and methods to translate the readable string into a value and the vice versa.

If an attribute has the data type DT_ENUM, the API will use the computer optimized representation of the values and expect a value of type T_LONG at the methods *getValue* and *setValue*. It is up to the client to transfer these values in the readable string representation. For performance reasons it will be wise for the client to cache the enumeration definition, otherwise a lot of client/server network traffic will be produced. If this attribute will be used as sorting criteria, also the value (of type T_LONG) will be used for sorting, not the string representation.

ASAM ODS servers allow the client to create or modify the application model using the OO-API. The new interface *EnumerationDefinition* also has some methods to modify the name or fixed name value pairs of the enumeration. These methods modify the application model and may only be used by superusers (nobody else is allowed to modify the application model).

A method *setEnumerationDefinition* at the *ApplicationAttribute* allows the client to set the enumeration of an certain attribute.

The methods *CreateEnumerationDefinition* and *RemoveEnumerationDefintion* at the interface *ApplicationStructure* allow the client to create a new or remove an existing object. Methods like *getEnumerationDefinition* and *listEnumerationDefinition* at the interface *ApplicationStructure* interface allow the client a general access to the enumerations.

For a detailed description of the methods to handle enumerations, see the document *ASAM ODS Version 5.0, Chapter 10, The OO-API, section 10.2.5, ApplicationAttribute and 10.2.16, EnumerationDefinition*.

Note: The enumerations defined by ASAM ODS should be handled the same way as the application specific enumerations.

The values of the enumeration items start with 0 and have no gap.

The following example shows the enumeration seq_rep_enum given in the base model.

ENUMID	ENUMNAME	ITEM	ITEMNAME
1013	seq_rep_enum	0	EXPLICIT
1013	seq_rep_enum	1	IMPLICIT_CONSTANT
1013	seq_rep_enum	2	IMPLICIT_LINEAR
1013	seq_rep_enum	3	IMPLICIT_SAW
1013	seq_rep_enum	4	RAW_LINEAR
1013	seq_rep_enum	5	RAW_POLYNOMIAL
1013	seq_rep_enum	6	FORMULA
1013	seq_rep_enum	7	EXTERNAL_COMPONENT
1013	seq_rep_enum	8	RAW_LINEAR_EXTERNAL
1013	seq_rep_enum	9	RAW_POLYNOMIAL_EXTERNAL
1013	seq_rep_enum	10	RAW_LINEAR_CALIBRATED
1013	seq_rep_enum	11	RAW_LINEAR_CALIBRATED_EXTERNAL

This table is extensible.

In the base model the names of the items are lower case, programmers normally use upper case names for the enumeration items, but it is up to the creator of the model what to use. It is important that the name of the item is case sensitive.

The new definition DT_ENUM was added to the data type enumeration. Also the definition DS_ENUM was added to store a sequence of enumerations in an attribute, e.g. values of AoLocalColumn.

ENUMERATION DATA TYPES

The new definitions DT_ENUM and DS_ENUM (for storage of a sequence of enumerations, e.g. values of the AoLocalColumn) were added to the data type enumeration:

```
/*
* The ASAM ODS data types.
* DT_xxx Basic data types.
* DS_xxx Sequence of basic data type.
* ||
* |+- T == Type, S == Sequences.
* +-- D == DataType.
*/
enum DataType {
DT_UNKNOWN,           // Unknown data type.
DT_STRING,            // String.
DT_SHORT,             // Short value (16 bit).
DT_FLOAT,             // Float value (32 bit).
DT_BOOLEAN,           // Boolean value.
DT_BYTE,              // Byte value (8 bit).
DT_LONG,              // Long value (32 bit).
DT_DOUBLE,            // Double precision float value (64 bit).
DT_LONGLONG,          // LongLong value (64 bit).
DT_ID,                // LongLong value (64 bit). Not used. DT_LONGLONG is used
                      //  instead.
DT_DATE,              // Date.
DT_BYTESTR,           // Bytestream.
DT_BLOB,              // Blob.
DT_COMPLEX,           // Complex value (32 bit each part).
DT_DCOMPLEX,          // Complex value (64 bit each part).
DS_STRING,            // String sequence.
DS_SHORT,             // Short sequence.
DS_FLOAT,             // Float sequence.
DS_BOOLEAN,           // Boolean sequence.
DS_BYTE,              // Byte sequence.
DS_LONG,              // Long sequence.
DS_DOUBLE,            // Double sequence.
DS_LONGLONG,          // LongLong sequence.
DS_COMPLEX,           // Complex sequence.
DS_DCOMPLEX,          // Double complex sequence.
DS_ID,                // LongLong sequence. Not used. DS_LONGLONG is used instead.
DS_DATE,              // Date sequence.
DS_BYTESTR,           // Bytestream sequence.
DT_EXTERNALREFERENCE, // External reference.
DS_EXTERNALREFERENCE, // Sequence of external reference.
DT_ENUM,              // The enumeration data type.
DS_ENUM               // The enumeration sequence data type.
};
```

IMPACT ON TS_UNION AND TS_UNIONSEQ

The unions TS_Union and TS_UnionSeq have as discriminator a value of type DataType. When the enumeration nv DataType is extended with the new definition DT_ENUM and DS_ENUM, the unions also need the extension for that new data types. For optimal computer performance the value of the enumeration will be given, the value is of type T_LONG, so the cases in the unions will be identical with the DT_LONG and DS_LONG cases.

This results in the following new definition:

```
/*
The Union definition for all data types.
*/
union TS_Union switch (DataType) {
case DT_STRING: T_STRING stringVal;
case DT_SHORT: T_SHORT shortVal;
case DT_FLOAT: T_FLOAT floatVal;
case DT_BYTE: T_BYTE byteVal;
case DT_BOOLEAN: T_BOOLEAN booleanVal;
case DT_LONG: T_LONG longVal;
case DT_DOUBLE: T_DOUBLE doubleVal;
case DT_LONGLONG: T_LONGLONG longlongVal;
case DT_COMPLEX: T_COMPLEX complexVal;
case DT_DCOMPLEX: T_DCOMPLEX dcomplexVal;
case DT_DATE: T_DATE dateVal;
case DT_BYTESTR: T_BYTESTR bytestrVal;
case DT_BLOB: T_BLOB blobVal;
case DS_STRING: S_STRING stringSeq;
case DS_SHORT: S_SHORT shortSeq;
case DS_FLOAT: S_FLOAT floatSeq;
case DS_BYTE: S_BYTE byteSeq;
case DS_BOOLEAN: S_BOOLEAN booleanSeq;
case DS_LONG: S_LONG longSeq;
case DS_DOUBLE: S_DOUBLE doubleSeq;
case DS_LONGLONG: S_LONGLONG longlongSeq;
case DS_COMPLEX: S_COMPLEX complexSeq;
case DS_DCOMPLEX: S_DCOMPLEX dcomplexSeq;
case DS_DATE: S_DATE dateSeq;
case DS_BYTESTR: S_BYTESTR bytestrSeq;
case DT_EXTERNALREFERENCE: T_ExternalReference extRefVal;
case DS_EXTERNALREFERENCE: S_ExternalReference extRefSeq;
case DT_ENUM: T_LONG enumVal;
case DS_ENUM: S_LONG enumSeq;
};
```

```
/*
Define a union with sequences of a certain type. Using this
union instead of sequence <TS_Union> gives much better
performance.
*/

union TS_UnionSeq switch (DataType) {
case DT_STRING: S_STRING stringVal;
case DT_SHORT: S_SHORT shortVal;
case DT_FLOAT: S_FLOAT floatVal;
case DT_BYTE: S_BYTE byteVal;
case DT_BOOLEAN: S_BOOLEAN booleanVal;
case DT_LONG: S_LONG longVal;
case DT_DOUBLE: S_DOUBLE doubleVal;
case DT_LONGLONG: S_LONGLONG longlongVal;
case DT_COMPLEX: S_COMPLEX complexVal;
case DT_DCOMPLEX: S_DCOMPLEX dcomplexVal;
case DT_DATE: S_DATE dateVal;
case DT_BYTESTR: S_BYTESTR bytestrVal;
case DT_BLOB: S_BLOB blobVal;
case DS_STRING: SS_STRING stringSeq;
case DS_SHORT: SS_SHORT shortSeq;
case DS_FLOAT: SS_FLOAT floatSeq;
case DS_BYTE: SS_BYTE byteSeq;
case DS_BOOLEAN: SS_BOOLEAN booleanSeq;
case DS_LONG: SS_LONG longSeq;
case DS_DOUBLE: SS_DOUBLE doubleSeq;
case DS_LONGLONG: SS_LONGLONG longlongSeq;
case DS_COMPLEX: SS_COMPLEX complexSeq;
case DS_DCOMPLEX: SS_DCOMPLEX dcomplexSeq;
case DS_DATE: SS_DATE dateSeq;
case DS_BYTESTR: SS_BYTESTR bytestrSeq;
case DT_EXTERNALREFERENCE: S_ExternalReference extRefVal;
case DS_EXTERNALREFERENCE: SS_ExternalReference extRefSeq;
case DT_ENUM: S_LONG enumVal;
case DS_ENUM: SS_LONG enumSeq;
};
```

Remark: The server and client should also accept DT_LONG or DS_LONG for compatibility reasons.

For a detailed description of Unions in ASAM ODS, see the document *ASAM ODS Version 5.0, Chapter 10, The OO-API, section 10.3.16, ASAM ODS UNIONS*.

IMPACT ON THE PHYSICAL STORAGE OF EARLIER VERSIONS

The definition of the physical storage for relational databases is not downward compatible, therefore, servers of previous versions will find values DT_ENUM or DS_ENUM in the column *ADTYPE* of the table *SVCATTR* and do not know how to handle these. So once the physical storage is upgraded to the ASAM ODS Version 5.0 using enumerations, the server must be exchanged.

When servers with the newer version work on previous versions of the physical storage, they will not find the table *SVCENUM*. It is recommended to check the existence of the table before this table will be used. If the table *SVCENUM* does not exist and the new methods of the interface *ApplicationStructure* or *EnumerationDefinition* are called, the exception AO_NOT_IMPLEMENTED should be thrown.

It is recommended that a server compliant to this definition should work properly even if the column *ENUMNAME* does not exist in the table *SVCATTR*.

3.2.7 SVCACLI

This table holds the security data for the instance protection. The syntax for creating this table is as follows:

```
create table svcacli
(
  USERGROUPID    integer NOT NULL,   /* Reference to a UserGroup instance */
  AID            integer NOT NULL,   /* Application element id */
  IID            integer NOT NULL,   /* Application instance id */
  RIGHTS         integer NOT NULL    /* Rights value (bit masked) */
);
```

The column *RIGHTS* holds the five basic rights Read, Update, Insert, Delete, and Grant as a bit masked value. If the respective bit is set, it means the respective right is granted. The following table shows the sequence of the rights:

Bit 0	Bit 1	Bit 2	Bit 3	Bit 4
Read	Update	Insert	Delete	Grant

The table looks like this:

Logical:	Group ID	Appl. Element ID	Instance ID	Rights
Database-column:	USERGROUPID	AID	IID	RIGHTS

3.2.8 SVCACLA

This table holds the security data for the attribute and element protection. The syntax for creating this table is as follows:

```
create table svcacla
(
  USERGROUPID    integer NOT NULL,   /* Reference to a UserGroup instance */
  AID            integer NOT NULL,   /* Application element id */
  AANAME         char(30),           /* Application attribute name */
  TYPE           char(5) NOT NULL,   /* Selector ("AA" or "AE") for rights on
                                        attribute or element */
  RIGHTS         integer NOT NULL    /* Rights value (bit masked) */
);
```

The column *TYPE* holds the selector for the security data, whether they are given for the Application Attribute (AA) or the Application Element (AE).

The column *RIGHTS* holds the five basic rights Read, Update, Insert, Delete, and Grant as a bit masked value. If the respective bit is set, it means the respective right is granted. The following table shows the sequence of the rights:

Bit 0	Bit 1	Bit 2	Bit 3	Bit 4
Read	Update	Insert	Delete	Grant

The table looks like this:

Logical:	Group ID	Appl. Element ID	Attr. Name	Type	Rights
Database-column:	USERGROUPID	AID	AANAME	TYPE	RIGHTS

3.2.9 SVCTPLI

This table holds the ACL templates for security. The syntax for creating this table is as follows:

```
create table svctpli
(
  USERGROUPID    integer NOT NULL,   /* Reference to a UserGroup instance */
  AID            integer NOT NULL,   /* Application element id */
  IID            integer NOT NULL,   /* Application instance id */
  REFAID         integer NOT NULL,   /* Referencing application element */
  RIGHTS         integer NOT NULL    /* Rights value (bit masked) */
);
```

The table looks like this:

Logical:	Group ID	Appl. Element ID	Instance ID	Ref. Appl-ID	Rights
Database-column:	USERGROUPID	AID	IID	REFAID	RIGHTS

If an entry in **SVCTPLI** is attached to an application element only, the InstanceID must be set to 0.

In the **SVCATTR** table a new column **ACLREF** with data type short-integer (values: 0/1) must be added . This flag tells the server which reference-attribute(s) must be resolved to find the ACL-Template that will be used while creating a new instance.

3.3 STORAGE OF THE ATTRIBUTE VALUES

3.3.1 STORAGE OF SINGLE VALUES

Most attributes are single value attributes. These values are stored directly in the column given in the table *svcattr*. The ASAM ODS data type corresponds with the data type of the database engine.

3.3.2 STORAGE OF BYTESTREAM VALUES

The values of the attributes with the data type T_BYTESTR are stored in a column of the database with the data type 'long raw'. The length of the bytestream corresponds with the length of the 'long raw' field.

3.3.3 STORAGE OF BLOB VALUES

The values of the attributes with the data type T_BLOB are stored in two columns of the table. The header of the Blob is stored in a column whose name is given in the column *dbcname* of the table *svcattr.* The database data type of this column is 'varchar'. The body of the Blob attribute is given in a separate column, with a fixed name, called 'Blob'. This column has the data type 'long raw'.

Only one attribute of the type T_BLOB for each element can be stored in the physical storage for relational databases.

3.3.4 STORAGE OF ARRAY VALUES AND OBJECT VALUES FOR ATTRIBUTES

In ASAM ODS Version 5.0 OO-API it is defined that attribute values may be an array of values or objects (DT_EXTERNALREFERENCE). In the specification of the physical storage there is no definition how to store these values, only the storage of the array values of the local column is defined.

Therefore, the values or the fields of the objects should be stored in a separate table. For each element there will be one table for all object or sequence attributes defined. The name of the table will be the name of the table from the element given in the table *svcent* concatenated with the extension "_ARRAY". The length of the complete table name **must not** exceed the maximum allowed length within the database. This means e.g. for Oracle data bases the length of <Table name> must not exceed 24 characters (the maximum table name is 30 characters minus the six characters "_ARRAY"). This table has columns for the instance element id (called IID), the ordinal number of the value (called ORD) and a column with the values for each attribute. The name of the column with the attribute values is given in the column *dbcname* of the table *svcattr*. The data type of the column with the values corresponds with the data type given in the column *adtype* of the table *svcattr*. If there are more then one sequence- or object-attribute-values available, the values are 'packed' in the table.

EXAMPLE:

TABLE: TEST_ARRAY

The Instance of Test with the Id 1 has two sequence attributes, a sequence of 4 double values, and a sequence with 2 long values. The Instance of Test with the Id 2 has two sequence attributes, a sequence of 2 double values, and a sequence with 4 long values.

```
IID    ORD     DOUBLE_SEQ     LONG_SEQ
1      1       1.0            2
1      2       2.0            4
1      3       3.0
1      4       4.0
2      1       1.0            2
2      2       2.0            4
2      3                      6
2      4                      8
```

The attribute values of objects (like the data types T_EXTERNALREFERENCE, T_COMPLEX, T_DCOMPLEX) have their own ordinal number for each field.

The ordinal number is starting with 0.

The object attribute values of type T_EXTERNALREFENCE, T_COMPLEX and T_DCOMPLEX are stored in this table. Each field of this object has his own ordinal number.

Type	Field	Ordinal Number	Field data type
T_EXTERNALREFERENCE	description	0	T_STRING
T_EXTERNALREFERENCE	mimeType	1	T_STRING
T_EXTERNALREFERENCE	location	2	T_STRING
T_COMPLEX	r	0	T_FLOAT
T_COMPLEX	I	1	T_FLOAT
T_DCOMPLEX	r	0	T_DOUBLE
T_DCOMPLEX	i	1	T_DOUBLE

The ASAM ODS RPC-API is not able to handle attributes with array values. A modification of the application model is required, so that there are no existing data storages with attributes with array values. As soon as the application model will be modified it cannot be accessed by the RPC-API anymore. There is no impact on all the existing data storages without this feature.

3.3.5 STORAGE OF ATTRIBUTE VALUE FLAGS

The ASAM ODS Version 5.0 OO-API allows the client to set value flags for attribute values like the value flags of the measurement data values. The following value flags are defined for the measurement data:

Abbreviation	Value	Description
AO_VF_VALID	0x01	Value is valid.
AO_VF_VISIBLE	0x02	The value has to be visualized.
AO_VF_UNMODIFIED	0x04	Value is not modified.
AO_VF_DEFINED	0x08	Value is defined. If the value in a value matrix is not available this bit is not set.

Therefore, an additional database-column (hidden on the interface) called 'ATTRIBUTEFLAGS' of type VARCHAR(255) is added, where each position in the string holds a hexadecimal number (0-9 & A-F) encoding the flags for each attribute, like the coding of the value flags. The attribute number *ATTRNR* of the table *SVCATTR* should be the index into this "array" of characters.

Example: element-name: Test, with attributes: ID, NAME, DESC, TESTBEGIN, TESTEND. The database column is ATTRIBUTEFLAGS. The value in ATTRIBUTEFLAGS could look like: "FFFFE". Here the attribute TESTEND is INVALID (value of FLAGS[4]=0xE). If the whole ATTRIBUTEFLAGS value is NULL or a zero-string, the whole instance is valid (compatibility to already stored instances). If the *FLAG* of the base attribute "ID" is set to INVALID, the whole instance is invalid (easy method to implement flags for instances).

Impact on ASAM ODS RPC-API : Check the numbering of column *ATTRNR* in the table *SVCATTR*. It may be required to modify the numbering starting with 0 and to be continuous. Check that there is no column with the name “ATTRIBUTEFLAGS”. If one of the two check fails this can be corrected and the tables *SVCENT* and *SVCATTR* can be modified so the ODS-Server with RPC-API can be run again. The column ATTRIBUTEFLAGS is hidden to the interface, so it will not be called in the table *SVCATTR*, therefore, a ODS-Server with RPC-API will not see this table.

3.3.6 STORAGE OF RELATIONS TO THE SUPERCLASS

The physical storage for relational databases is not able to store a relation to a super class. A relation from an application element of type *AoTestEquipmentPart* has a relation to the parent *AoTestEquipmentAbstract*. The "Abstract"-elements are not part of the physical storage. It is not possible to store these kind of relations in the physical storage because the column *faid* of the table *svcattr* expects an application element Id. There is no application element Id of the "Abstract"-element and there may be a lot of application element Ids of the derived application elements.

Therefore, an own entry in the table *svcattr* for each possible parent with all the same base relation names in the column *baName* of that table should be used. Then there will be a change of the base model, and base relations may be used more than once. The client has to take care which relation has a valid value.

The ODS-Server with RPC-API will deliver both relation attributes with the same base name, the client has to take care which attribute must be used while reading the instances. The ODS-Server with OO-API hides this information for the client when the object oriented methods are used and the server has to take care. When the client uses the structure oriented function the client has to take care, like the client must do it with the RPC-API. The ODS-Server with RPC-API will have no problems to serve this information.

3.4 THE MIXED-MODE-SERVER

As the number of measurement data to be stored increased rapidly during the last years, there is a strong demand to store these data in separate files outside the database. Currently, mass data often are stored as blobs in the RDB. The disadvantage is obvious: searching for specific data often is extremely time-consuming due to the restrictions (e.g. Oracle has to process blobs within a database).

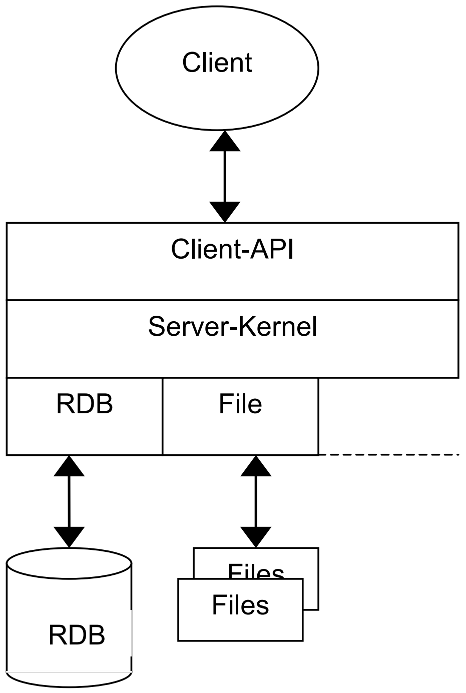
The solution is to hold mass data in separate files and the description of the data within the database. This approach automatically leads to a Mixed-Mode-Server.

A Mixed-Mode-Server must fulfill the following requirements:

- Access to RDB
- Storage of mass data in separate files outside the database
- Transparency to API
- Asynchronous access

3.4.1 MODEL OF MIXED-MODE-SERVER

The following figure shows the model of a Mixed-Mode-Server:



The data stored in separate files will be restricted to mass data only (e.g. measurement data leading to problems within the RDB).

3.4.2 USE CASES

The following tables show some use cases of measurement data gained:

HOMOGENOUS SUBMATRIX

Homogenous submatrices with measurements taken from different channels at the same time intervals:

Time interval	Channel 1	Channel 2	Channel 3
1	Val 1	Val 1	Val 1
2	Val 2	Val 2	Val 2
3	Val 3	Val 3	Val 3
...	....	....	....

LOCAL COLUMN

Local Column with sequential measurements at the same time interval:

Time interval	Channel 1
1	Val 1
2	Val 2
3	Val 3
...	....

VALUE MATRIX

A lot of measurements in blocks from different channels at the same time interval:

Time interval	Channel 1	Channel 1	Channel 2	Channel 2	Channel 3	Channel 3
1	Val 1	Val 2	Val 1	Val 2	Val 1	Val 2
2	Val 3	Val 4	Val 3	Val 4	Val 3	Val 4
3	Val 5	Val 6	Val 5	Val 6	Val 5	Val 6
....	....	....	....	....	....	....

The parameters needed for accessing the measurement data in respect to the above mentioned three use cases are:

- Number of measurement values
- Start Offset
- Type (implicit length)

- Number of block values
- Skip

The following table shows the correlation of the parameters and the three use cases:

	Use Case 1	Use Case 2	Use Case 3
Number of measurement values	n	n	n
Start Offset	0	0	0
Type (implicit length)	F	F	F
Number of block values	1	n	2
Skip	1,2,3,...	N/A	6

A fourth use case was also discussed:

A lot of channels measured block-wise at different time intervals leading to a table similar to the following:

Time interval	Channel 1	Channel 2	Channel 3	...
1	Val 1	Val 1	Val 1	....
2		Val 2		
3	Val 2	Val 3	Val 2	....
4	Val 3			
5	Val 4	Val 4	Val 3	....
6			Val 4	
....	....	....	....	....

It was decided not to specify this case for a Mixed-Mode-Server. There should be converters to solve the problems with this use case.

3.4.3 APPROACH FOR SPECIFYING MIXED-MODE-SERVER

It was decided to use the binary *component* from ATF to specify the format for separate files in a Mixed-Mode-Server environment (see *ASAM ODS Version 5.0, Chapter 5, The ASAM Transport Format Classic (ATF/CLA), section 5.7.22, Structure of* ***component***).

The binary file must fulfill the following requirements:

➢ There must be an optional mini header for general information. This header will not be evaluated by the server.

➢ The stored binary file will hold the mass data, while the descriptive data are stored in the RDB.

MAPPING OF REQUIRED ATTRIBUTES FROM ATF

The following table shows the attributes from the ATF description that will be used for specifying the binary file within a Mixed-Mode-Server environment:

Attribute	Description
Identifier	Filename (URL)
Type-Specification	Data type and byte order
Length specification	Number of values
IniOffset	Start Offset
Blocksize	Size of a row of the block, also defining the offset for the next value (if local column holds only one value)
ValOffsets	Offset from begin of block to respective value
ValPerBlock	Number of values/block

APPROACH FOR STORING THE ATTRIBUTES

To store the requested attributes, the base element *AoExternalComponent* was added, and a change of *AoLocalColumn* was necessary. The following listing shows the base element as well as the related attributes and type definitions, described with STEP EXPRESS (see also *ASAM ODS Version 5.0, Chapter 4, The ODS Base Model*):

```
ENTITY AoExternalComponent (* BID=40 *)
 SUBTYPE OF (asam_base_entity);
  filename_url       : STRING;
  start_offset       : INTEGER;
  blocksize          : INTEGER;
  valuesperblock     : INTEGER;
  value_offsets      : LIST [1:?] OF INTEGER;
  flags_filename_url : STRING;
  flags_start_offset : INTEGER;
UNIQUE
  UR1: SELF\asam_base_entity.id;
END_ENTITY;

TYPE seq_rep_enum = ENUMERATION OF (
   explicit,
   implicit_constant,
   implicit_linear,
   implicit_saw,
   raw_linear,
   raw_polynomial,
   formula,
   external_component       (* <--- NEW *)
   (* extendible!! *)
 );

ENTITY AoLocalColumn (* BID = 39 *)
 SUBTYPE OF (asam_base_entity);
  flags                   : OPTIONAL LIST [1:?] OF t_short;
  global_flag             : t_short;
  independent             : BOOLEAN;
  minimum                 : OPTIONAL t_double;
  maximum                 : OPTIONAL t_double;
  sequence_representation : seq_rep_enum;
  generation_parameters   : OPTIONAL LIST [1:?] OF t_double;
  values                  : OPTIONAL value_sequence;
  external_component      : OPTIONAL AoExternalComponent;
INVERSE
  submatrix            : AoSubmatrix for local_columns;
  measurement_quantity : AoMeasurementQuantity for local_columns;
 (* existent rules omitted here *)
 (* rules to be added:
     If attribute "external_component" has a value, then the
     attributes "generation_parameters" and "values" must be
     empty;
     if one of these has a value then "external_component"
     must be empty.
     If attribute "external_component" has a value, then the
     attribute "flags" must be empty.
  *)
END_ENTITY;
```

IMPACT ON ODS-SECURITY

The impact on the ODS-Security will be the following:

- The rights of the measurement will be inherited.
- The storage of the binary file must be done via the API of ASAM ODS (copy).

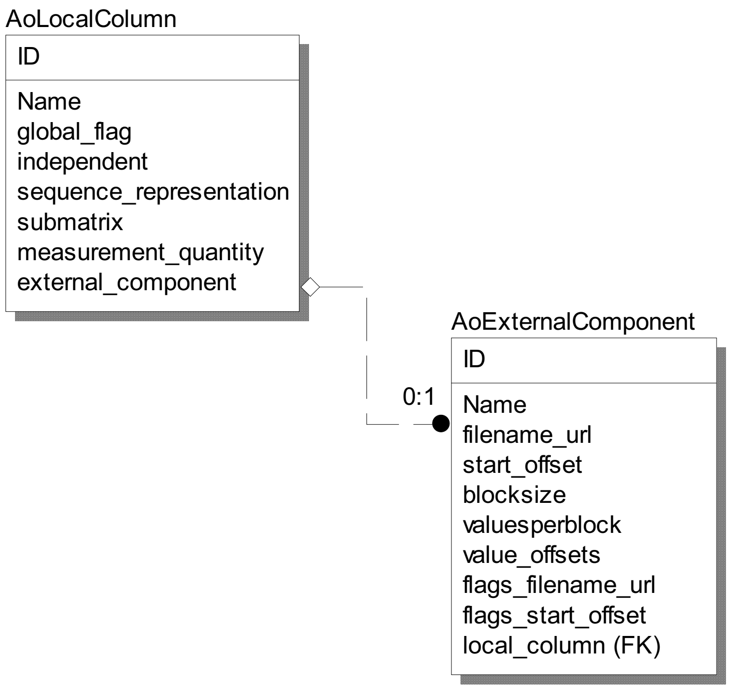
MODE-SWITCHING OF MIXED-MODE-SERVER

To enforce the writing of an external file or the storage within the database of the server, an additional parameter is needed. This parameter must be set in AoSession as context variable (see *ASAM ODS Version 5.0, Chapter 10, The ASAM ODS Version 5.0 OO-API), section AoSession_SetContext*). The value will be ***write_mode*** with these possible values: ***database (default) / file.***

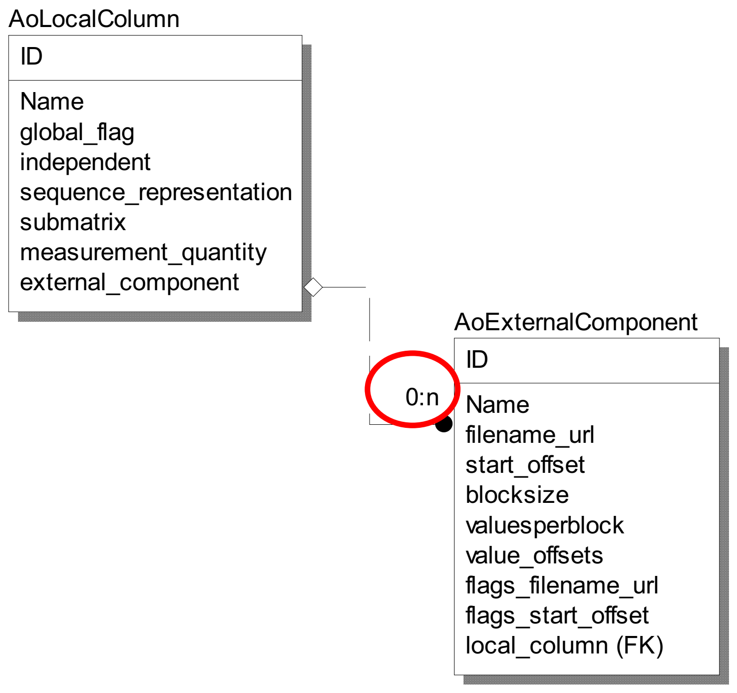
3.4.4 SEGMENTATION WITHIN THE MIXED-MODE-SERVER

There is a need for the possibility to split one local column into several pieces when using external components. This *section* shows one method to include segmentation of an external component file. Segmentation is used to split one big binary "external-component-file" into several smaller ones to improve performance, concurrency and file-handling.

Therefore, the data model needs to be extended. The current version of AoLocalColumn-AoExternalComponent looks like this:



This new approach including segmentation looks like this:



Changes in the basemodel:

Relationship changed from "zero or one" to "zero, one or many".

Each instance of the application element LocalColumn may have one or more child-instances of ExternalComponent.

PHYSICAL IMPLEMENTATION (RDBMS)

Each instance in ExternalComponent must be numbered with an appropriate attribute *segment_nr*. The first segment starts with the value 1. The sequence of *segment_nr* must start with 1 for <u>each</u> LocalColumn instance. *segment_nr* is a mandatory attribute and must hold a value greater or equal 1.

PARAMETER FOR DEFINING THE FILE SIZE OF EACH SEGMENT

A new server-side parameter must be introduced that can be modified by the client. This parameter is called EXT_COMP_SEGSIZE (DT_LONG) and defines the maximum file size of new segments in bytes. When the client changes the value of this parameter, all following write operations must use the new definition. The default value of this parameter should be set at server-start-up-time to an appropriate value by using a server-side configuration mechanism (e.g. start-up-file, or start-up-parameters).

3.4.5 USE-CASES

CREATE A NEW LOCALCOLUM WITH INITIAL VALUES

- Create new external-component file with values.
- Create additional segments, depending on the number of values to write and the file size of the external-component-file.

APPEND VALUES TO A LOCALCOLUMN

- Check number of values and file-size in the existing (last) external-component-file and decide whether to append the values to the file or to create a new file (segment).

INSERT VALUES IN A LOCALCOLUMN

- Check number of values and file size in the existing external-component-file and decide whether the file can hold the total number of values after inserting the new values into one file.
- If the values can be inserted into the existing file without splitting, the file must be read completely and written back including the new values.
- If the file must be splitted into one or more segments, the affected file (segment) must be read completely and the new files must be written back to disk.

MODIFY VALUES IN A LOCALCOLUMN

- For numerical data type (fixed length), direct update in the file is possible without moving data.
- For strings, byte-streams and blobs the file must be read and written back according to the above described Use Case "Insert values in a LocalColumn".

DELETE VALUES IN A LOCALCOLUM

- The values in the affected files are deleted.
- If all values are deleted inside one segment, the file and ExternalComponent-instance and LocalColumn instance in the database must be deleted.

DELETE WHOLE LOCALCOLUMN

- Check, if the external-component-file holds values of other LocalColumns, too.
- If this is the case, the whole file may be read and written back completely to remove the values physically. The instance in LocalColumn and ExternalComponent must be deleted in the database.
- Otherwise, the whole external-component-file must be deleted.

3.4.6 CONSIDERATIONS FOR MIXED-MODE SERVER IMPLEMENTATIONS

- The server must insure the consistency of the data after any write operation on the external-component file. E.g. if the client sends a delete request on some values of one LocalColumn and not on the remaining LocalColumns inside one partial matrix, the server must detect this inconsistency and must be able to do a "rollback" (keep a file copy or detect it before writing, etc.).
- The server must lock the file during modification (Shared lock – no write access on the involved files, only read access). The server must hold a private working copy of the file and place an exclusive lock on the involved files during the "commit"-phase.
- The server must support some error recovery on start-up after a system crash.

3.5 REVISION HISTORY

Date Editor	Changes
2003-06 P. Voltmann	Description of Mixed-Mode-Server added Description of segmentation within Mixed-Mode-Server added Description of table SVCENUM added Description of enumerations extended (affecting SVCATTR) Description of additional column ENUMNAME in SVCATTR added Description of data type conversion added Several descriptions in SVCACLA and SVCACLI added Description of generation_parameters within the physical storage added (affecting SVCVAL)
2003-10-11 R. Bartz	Several errors have been fixed and explanations have been changed to make things clearer Duplicate sections have been deleted
2003-10-17	The FTR meeting agreed to the current text with one modification required
2003-11-21 R. Bartz	Some changes related to data types have been introduced
2003-12 R. Bartz	Ambiguity of ADTYPE resolved by introducing ENUMID Minor textual changes have been introduced
2003-12-30 R. Bartz	The **Release** version has been created

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 4
BASE MODEL

Version 28

Association for **S**tandardization of
Automation and **M**easuring Systems

Dated: 30.09.2004

Status of Document

Reference:	ASAM ODS Version 5.0 Base Model
Date:	30.09.2004
Author:	Dr. Helmut Helpenstein, National Instruments; Karst Schaap, HighQSoft; Gerald Sammer, AVL
Type:	Specification
Doc-ID:	ASAM_ODS_50_CH04_Base_Model.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 (odsftr@asam.net) to make sure this issue will be addressed within the next review cycle.

Contents

4 THE BASE MODEL 4-5

4.1 INTRODUCTION TO THE BASE MODEL 4-5
4.1.1 OVERVIEW 4-5
4.1.2 PRINCIPLE OF DERIVATION 4-6
4.1.3 GENERAL REMARKS 4-6
4.2 DESCRIPTION OF BASE ELEMENTS FOR ENVIRONMENT 4-7
4.2.1 AOENVIRONMENT (BID = 1) 4-7
4.2.2 AONAMEMAP (BID = 46) 4-9
4.2.3 AOATTRIBUTEMAP (BID = 47) 4-10
4.3 DESCRIPTION OF BASE ELEMENTS FOR DIMENSIONS AND UNITS 4-11
4.3.1 OVERVIEW 4-11
4.3.2 INFORMATION ABOUT QUANTITIES AND UNITS IN ASAM ODS 4-11
4.3.3 AOQUANTITY (BID = 11) 4-13
4.3.4 AOUNIT (BID = 13) 4-15
4.3.5 AOPHYSICALDIMENSION (BID = 15) 4-17
4.3.6 AOQUANTITYGROUP (BID = 12) 4-19
4.3.7 AOUNITGROUP (BID = 14) 4-20
4.3.8 CONVERSION OF UNITS 4-21
4.4 DESCRIPTION OF BASE ELEMENTS FOR MEASUREMENTS 4-23
4.4.1 OVERVIEW 4-23
4.4.2 AOMEASUREMENT (BID = 3) 4-24
4.4.3 AOMEASUREMENTQUANTITY (BID = 4) 4-26
4.4.4 SUBMATRICES AND THE VALUE MATRIX 4-28
4.4.5 AOSUBMATRIX (BID = 38) 4-30
4.4.6 AOLOCALCOLUMN (BID = 39) 4-31
4.4.7 AOEXTERNALCOMPONENT (BID = 40) 4-35
4.5 DESCRIPTION OF THE BASE ELEMENTS FOR ADMINISTRATION 4-37
4.5.1 AOTEST (BID = 36) 4-37
4.5.2 AOSUBTEST (BID = 2) 4-38
4.6 DESCRIPTION OF THE BASE ELEMENTS FOR DESCRIPTIVE DATA 4-39
4.6.1 AOUNITUNDERTEST (BID = 21) 4-40
4.6.2 AOUNITUNDERTESTPART (BID = 22) 4-41
4.6.3 AOTESTSEQUENCE (BID = 25) 4-42
4.6.4 AOTESTSEQUENCEPART (BID = 26) 4-43
4.6.5 AOTESTEQUIPMENT (BID = 23) 4-44
4.6.6 AOTESTEQUIPMENTPART (BID = 24) 4-45
4.6.7 AOTESTDEVICE (BID = 37) 4-46
4.7 DESCRIPTION OF THE BASE ELEMENTS FOR SECURITY 4-47

4.7.1 AOUSER (BID = 34) 4-47
4.7.2 AOUSERGROUP (BID = 35) 4-48
4.8 DESCRIPTION OF THE BASE ELEMENTS FOR OTHER DATA 4-49
4.8.1 AOANY (BID = 0) 4-49
4.8.2 AOLOG (BID = 43) 4-50
4.8.3 PARAMETERS AND PARAMETER SETS 4-51
4.8.4 AOPARAMETER (BID = 44) 4-55
4.8.5 AOPARAMETERSET (BID = 45) 4-56
4.9 FORMAL DESCRIPTION OF THE ASAM ODS BASE MODEL 4-57
4.9.1 WHY STANDARDIZED REPRESENTATION 4-57
4.9.2 METAMODEL DESCRIPTION IN ISO 10303 4-58
4.9.3 BASE MODEL REPRESENTATION IN EXPRESS 4-60
4.9.4 BASE MODEL REPRESENTATION IN EXPRESS-G 4-96
4.10 REVISION HISTORY 4-107

Scope

This document describes the ASAM ODS Base Model 28 for ASAM ODS Version 5.0

Intended Audience

This document is intended for implementers of ASAM ODS Version 5.0. It shall be used as a reference on how the base model is built and which base elements, base attributes, relations etc. are available.

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

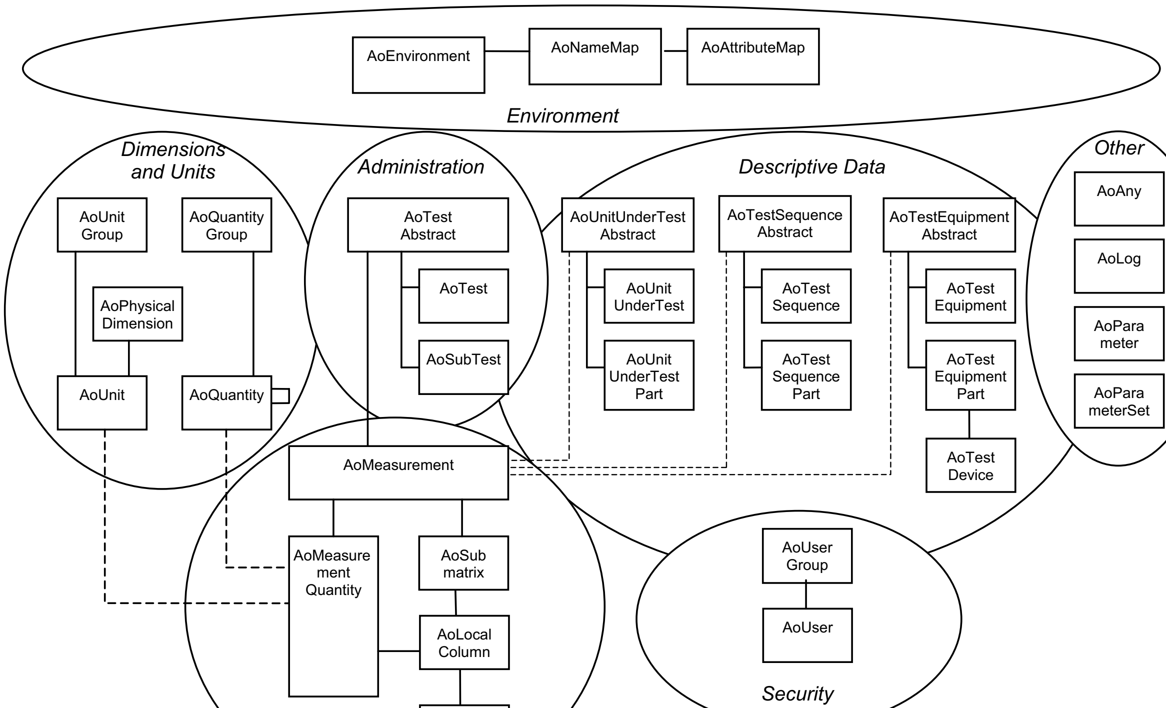
- ISO 10303-11: STEP EXPRESS
- Object Management Group (OMG): www.omg.org
- Common Object Request Broker Architecture (CORBA): www.corba.org
- IEEE 754: IEEE Standard for Binary Floating-Point Arithmetic, 1985
- ISO 8601: Date Formats
- Extensible Markup Language (XML): www.w3.org/xml

4 THE BASE MODEL

4.1 INTRODUCTION TO THE BASE MODEL

4.1.1 OVERVIEW

The following figure shows the base model of ASAM ODS:



The base model incorporates the following base elements, from which the application elements may be derived. These elements are grouped according to their usage within automation and measurement systems.

4.1.2 PRINCIPLE OF DERIVATION

Any ASAM ODS model should be able to carry measurement data together with the testing context and administration of the measurements. Since such data appear in many companies or divisions they cannot be mapped to the same model. On the other hand it has been found necessary to exchange data between different companies or divisions or between different applications.

To enable an exchange of data between various repositories the data models in these repositories must not be too different, even if differences are needed to comply with the special needs of certain applications or companies.

In ASAM ODS this contrast was dealt with by creating one base model from which all application-specific data models can be derived. Such derivation includes:

- subtyping (create a new type from an existing one (the supertype) and adding more properties or attributes)
- inheritance (a subtype inherits automatically all properties and attributes of the supertype)
- rules (the supertype provides rules to which the subtype must conform)

The derived model is usually typical for an application, therefore it is called application model, its entities are called application elements while the entities of the base model are called base elements.

4.1.3 GENERAL REMARKS

Characterizing an attribute or relation as OPTIONAL means both,

- an application model may omit this attribute.
- if an application model has specified this attribute then instances may have values for this attribute or may leave it empty.

All other attributes or relations are required. This means

- they must be included in the application model, and
- the instances must provide a value for them.
- In case the attribute or relation is a SET[0,..] or LIST[0,..], it may happen that the list or set is empty.

4.2 DESCRIPTION OF BASE ELEMENTS FOR ENVIRONMENT

4.2.1 AOENVIRONMENT (BID = 1)

The following table describes the base attributes of AoEnvironment:

Base Attributes	**Meaning**
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the ASAM ODS server
description	DT_STRING OPTIONAL Describing text for the ASAM ODS server
version	DT_STRING OPTIONAL Version of the ASAM ODS server
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs. Should not be used here because inheritance is not useful for application elements of type AoEnvironment
meaning_of_aliases	DT_STRING LIST[0:?]
max_test_level	DT_LONG OPTIONAL
base_model_version	DT_STRING OPTIONAL schema name (e.g. 'asam27')
application_model_type	DT_STRING OPTIONAL may contain many names, comma-separated
application_model_version	DT_STRING OPTIONAL any operator-supplied name
Relations:	
entity_mapping	INFO_FROM SET[0:?] OF AoNameMap

tests	CHILD SET[0:?] OF AoTest
uuts	CHILD SET[0:?] OF AoUnitUnderTest
equipments	CHILD SET[0:?] OF AoTestEquipment
sequences	CHILD SET[0:?] OF AoTestSequence

Of this entity only one instance may occur in any data storage.

Only instances of the top entities of the following 4 hierarchical trees are collected in the sets:

- AoTest
- AoUnitUnderTest
- AoTestEquipment
- AoTestSequence

In “entity_mapping” all maps are to be collected. The set contains one AoNameMap per entity. An “alias_index” in the application software allows to use the appropriate “alias_name” from the given lists. In this way language versions can be switched easily.

For each position in the lists of alias names a string should be provided in “meaning_of_aliases"; this allows for name mapping to different languages.

The attribute "max_test_level" shows the number of levels in the test hierarchy.

4.2.2 AONAMEMAP (BID = 46)

The following table describes the base attributes of AoNameMap:

Base Attributes	**Meaning**
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
entity_name	DT_STRING
alias_names	DT_STRING LIST[0:?]
Relations:	
environment	INFO_TO AoEnvironment INVERSE FOR entity_mapping
attribute_mapping	CHILD SET[0:?] OF AoAttributeMap

Within the instances of “AoNameMap” all entities of the application model (the “application elements”) are stored together with their relation to the corresponding base entities. It is possible to build a list of alias names. Additionally a list with the “attribute maps” for this application element is stored.

For the entity name any number of alias names (e.g. for language versions) may be specified. The list allows different language version switched by the application software.

The length of the list shall be equal to the list "meaning_of_aliases" in AoEnvironment.

For each entity a name mapping may be given.

For each attribute of the entity an attribute mapping may be given.

4.2.3 AOATTRIBUTEMAP (BID = 47)

The following table describes the base attributes of AoAttributeMap:

Base Attributes	Meaning
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance, only unique within its predecessor in a hierarchy
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
attribute_name	DT_STRING
alias_names	DT_STRING LIST[0:?]
Relations:	
name_mapping	FATHER AoNameMap INVERSE FOR attribute_mapping

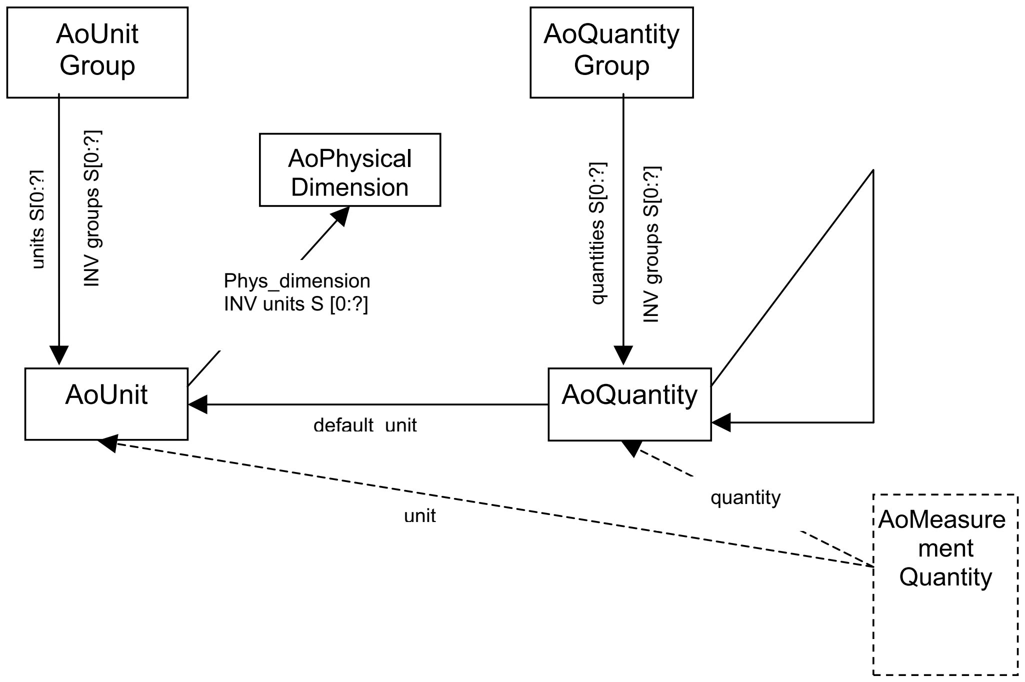
Each AoAttributeMap contains one attribute of an application element respectively. If an attribute was derived from a base attribute, the corresponding relation is also stored. It is further possible to build a list of alias names and to define a unit and a relation to a quantity.

Each instance of AoAttributeMap specifies any number of alias names for the attribute "attribute_name". The length of the list of alias names shall be equal to the length of that list in AoNameMap.

4.3 DESCRIPTION OF BASE ELEMENTS FOR DIMENSIONS AND UNITS

4.3.1 OVERVIEW

The following figure shows those base attributes which build relations between the base elements of the dimension/unit family. The arrows show the direction of the attributes, the text at the arrows shows the attribute name and the cardinality. The word “INV” precedes the same declarations for the inverse attributes (opposite to the direction of the arrow). S[a:b] marks a set with at least *a* and at most *b* elements. The question mark “?” means “any number”.



4.3.2 INFORMATION ABOUT QUANTITIES AND UNITS IN ASAM ODS

The definition and the usage of quantities and their names:

In measurement and test systems used today various information is coded in quantity names, which are often additionally restricted to e.g. 8 characters.

Such information may be:

- the name of the physical quantity itself (strength, time, concentration)
- the location where such a quantity typically is related to (jointed shaft in the front, on the left)
- the processing mode which is typically used to determine such a quantity from other quantities(averaged, filtered).

Such coding of information makes it practically impossible to automatically let the computer process the quantities on the basis of partial information (e.g. list all quantities which describe the characteristics of an object in the front on the left). Moreover there is often no distinction made between the description of a quantity and the description of the configuration established and the results obtained when measuring such a quantity.

Another feature which is demanded for quantities is the requirement to use the same names for standard quantities in the whole environment; for this purpose the names should be put into a list (as a catalog for standard quantity names).

On the other hand it does not make sense to define a global name for a quantity which is measured only once.

The use of quantities within ASAM ODS:

Quantities in ASAM ODS form a list of things that may be measured; they are ordered in a hierarchical relation.

Each of the "quantities" may be used in two ways: in a "measurement quantity" belonging to a "measurement", and in the hierarchy of "quantities" serving as predecessor. Note that ASAM ODS distinguishes between a quantity itself (as a neutral descripton) and the subject of a measurement. Whenever the measurement of a quantity like force, pressure, ... shall be initiated, a "measurement quantity" has to be instantiated and connected (by setting a reference) to a "quantity" first. The "measurement" only knows about its "measurement quantities". Each "measurement quantity" points to a "quantity" and thus provides information on what is currently measured; it refers to a quantity and receives from it several parameters, like its physical dimension and a default unit. Studying the attributes and relations of "AoMeasurement", "AoMeasurementQuantity" and "AoQuantity" will further clarify their interdependence.

The highest quantities in the hierarchy, i.e. those which do not have a predecessor, must be physical quantities to which a unit can be attached. In a normal case this would be a physical feature (e.g. force) rather than a location, a substance or the like (e.g. FL or CO). Thus it is guaranteed that every "quantity" possesses a unit and that it may be used in a "measurement quantity" (in this sense "dimensionless" or "[-]" is also seen as a unit).

It should be noted that the quantity hierarchy does NOT rely on the naming conventions but on the references of the instances of "quantity" to each other; these references (successor, predecessor) are embedded in the ASAM ODS base model.

An abbreviation can be attached to every quantity for the purpose of quick identification; it must be unique.

"Quantities" can be combined in "quantity groups" and one "quantity" can be member of many groups. This allows to group "quantities" according to application specific aspects (examples of "quantity groups": set values, temperatures, exhaust gas values).

Every "quantity" may be used one or more times in a measurement. If the "quantity" is used once in a single measurement, it is identified through the measurement and the connected measurement quantity names. An additional "measurement"-specific name may be defined. As a default for that "measurement quantity name" the "default measurement quantity name" (attribute “default_mq_name”) of the "quantity" should be used. If a "quantity” is used repeatedly in a single measurement, a different “measurement quantity name” must be given to every “measurement quantity”, to which the “quantity” is attached.

In that way local quantities (with respect to one "measurement") may be defined. For example if three temperatures, which are of no significance to other measurements, should be measured, then the referenced "quantity" is T and "measurement quantity names” may be generated like T1, T2 and T3. The “quantity” T has the function of a quantity type, which for instance supplies the unit.

4.3.3 AOQUANTITY (BID = 11)

The following table describes the base attributes of AoQuantity:

Base Attributes	Meaning
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
quantity_classification	quantity_class_enum OPTIONAL one of the enumeration values „measured“ or „state“; default value is "measured"
default_mq_name	DT_STRING Default name for a measurement quantity, to be used if no own name has been explicitly attached to the measurement quantity.
default_datatype	datatype_enum Default format in which the data is stored
default_rank	DT_LONG Default rank of a tensor, number of value dimensions
default_dimension	DT_LONG LIST[0:?] Default number of values for each rank
default_type_size	DT_LONG Default length limit of a value, for example the maximum length of a string or a bytestream
Relations:	
default_unit	INFO_TO AoUnit

successors	CHILD SET[0:?] OF AoQuantity
groups	INFO_REL SET[0:?] OF AoQuantityGroup INVERSE FOR quantities
measurement_quantities	INFO_FROM SET[0:?] OF AoMeasurementQuantity INVERSE FOR quantity
predecessor	FATHER SET[0:1] OF AoQuantity INVERSE FOR successors

A “quantity” is a named test variable with its features like
– unit, physical meaning
– data type of the value (real, integer, boolean, string, matrix, and others)

A quantity is not a measurement channel, it rather describes a physical phenomenon.

EXAMPLE: **4.3.3.1 QUANTITIES**

- torque (Real)
- time (Real)
- estimated temperature range {"cold", "warm", "hot”}
- ignition on (Boolean)
- ignition characteristic curve (which is applicable for a work point) (matrix of Real values).

A quantity may be used by many “measurement quantities“, even if these belong to the same “measurement“.

The quantities may be ordered in a hierarchical relationship to one another (using the successor/predecessor attribute), and there might occur quantities which are never assigned to a “measurement quantity“.

EXAMPLE: **QUANTITY HIERARCHY**

Quantity name	Successors	Predecessor	Measurement Quantity
m	CO	-	no
CO	CO before, CO after	m	no
CO before	-	CO	yes
CO after	-	CO	yes

All 4 are instances of the same application element (e.g. “quantity“), only “CO before“ and “CO after“ are used by measurement quantities, which get their default values for names, units, etc. from these quantities.

Please note that there are quantities that do not have a unit, e.g. strings, boolean quantities, percentages, etc.; for such purposes usually one “empty“ unit (e.g. with name “-“) shall be provided.

4.3.4 AOUNIT (BID = 13)

The following table describes the base attributes of AoUnit:

Base Attributes	Meaning
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
factor	DT_DOUBLE Factor to get the SI unit
offset	DT_DOUBLE Offset to get the SI unit
Relations:	
phys_dimension	INFO_TO AoPhysicalDimension
groups	INFO_REL SET[0:?] OF AoUnitGroup INVERSE FOR units
quantities	INFO_FROM SET[0:?] OF AoQuantity INVERSE FOR default_unit

factor is the (value using SI unit) when (value using this unit) = 1
offset is the (value using SI unit) when (value using this unit) = 0

(value using this unit) multiplied by (factor) = (value using SI unit)
(value using this unit) plus (offset) = (value using SI unit)
(value using this unit) multiplied by (factor) then plus (offset) = (value using SI unit)

EXAMPLE:	CALCULATING SI VALUE FROM GIVEN NON-SI VALUE	
	(4200) 1/min = (4200 * factor) 1/s = (70) 1/s	[factor=0.016667]
	(20) deg.C = (20 + offset) K = (293.15) K	[offset=273.15]
	(68) deg.F = (68 * factor + offset) K = (293,15) K	[factor=0.55555] [offset=255.372]
	(200) mm = (200 * factor) m = 0.2 m	[factor=0.001]

This entity specifies in which unit a measurement quantity has been measured. While the Physical Dimension is constant (e.g. mass) the units may vary (e.g. "g", "mg", "kg", "lbs"). "Units", which refer to the same "Physical Dimension", can be converted to each other by means of “Unit Offset” and “Unit Factor”. Two units may refer to the same physical dimension, only if the conversion between these units makes sense.

4.3.5 AOPHYSICALDIMENSION (BID = 15)

The following table describes the base attributes of AoPhysicalDimension:

Base Attributes	Meaning
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
lenght_exp	DT_LONG nominator of exponent for length
mass_exp	DT_LONG nominator of exponent for mass
time_exp	DT_LONG nominator of exponent for time
current_exp	DT_LONG nominator of exponent for electric current
temperature_exp	DT_LONG nominator of exponent for temperature
molar_amount_exp	DT_LONG nominator of exponent for molar amount
luminous_intensity_exp	DT_LONG nominator of exponent for light
lenght_exp_den	DT_LONG OPTIONAL denominator (length)
mass_exp_den	DT_LONG OPTIONAL denominator (mass)
time_exp_den	DT_LONG OPTIONAL

	denominator (time)
current_exp_den	DT_LONG OPTIONAL denominator (electric current)
temperature_exp_den	DT_LONG OPTIONAL denominator (temperature)
molar_amount_exp_den	DT_LONG OPTIONAL denominator (molar amount)
luminous_intensity_exp_den	DT_LONG OPTIONAL denominator (light)
Relations:	
units	INFO_FROM SET[0:?] OF AoUnit INVERSE FOR phys_dimension

A "physical dimension" is represented by the seven dimensional exponents of the SI base dimensions length, mass, time, temperature, current, molar amount, light intensity (measured in SI base units m, kg, s, K, A, Mol, cd). Usually many of the exponents are zero, particularly the dimensionless units (e.g. "%") have all exponents=0.

Please note that several physical dimensions may exist that have the same set of exponents.

EXAMPLE: **SIMILAR PHYSICAL UNITS**

Following physical dimensions may relate to the exponent set “length^2 * mass^1 * time^{-2}”:

- “energy“, used by units “Nm“, “kWh“,...
- “torque“, used by units “Nm“, “Nmm“,...

Following physical dimensions may relate to an exponent set of plain zeros:

- “proportion” used by units “%”, “ppm”,...
- “angle” used by units “°”, ”rad”,...

In most cases dimensional exponents are integers, i.e. their denominators are =1. This is automatically assumed if denominators are omitted.

4.3.6 AoQuantityGroup (BID = 12)

The following table describes the base attributes of AoQuantityGroup:

Base Attributes	**Meaning**
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
Relations:	
quantities	INFO_REL SET[0:?] OF AoQuantity

A "quantity group" describes the feature of a "quantity" set. One "quantity" may belong to many “quantity groups”; through that a net of relationships may be created.

EXAMPLE: **GROUP ASSIGNMENT OF QUANTITIES**

The quantity "F.FL" belongs, as indicated in its hierarchical name structure, to the forces ("F") and to a position “front left” ("FL").
It may additionally belong to the quantity groups: "length force", "calculated quantity", "filtered quantity" and “front left phenomena“.

4.3.7 AoUnitGroup (BID = 14)

The following table describes the base attributes of AoUnitGroup:

Base Attributes	**Meaning**
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
Relations:	
units	INFO_REL SET[0:?] OF AoUnit

AoUnitGroup allows grouping of "units" according to free criteria.

Examples: "MKS", "USA", "outdated".

4.3.8 CONVERSION OF UNITS

Every local unit is presentable through the product $\prod_{i=1}^{7} \text{SI-BU}_i^{\text{Expn}_i}$ of SI units, where the SI base units (SI-BU) {length in *m,* mass in *kg,* time in *s,* current in *A,* temperature in *K,* molar amount in *mol* and luminous intensity in *cd*} are supplied with a signed (in most cases integer) exponent (Expn), for example: acceleration in *m* s^{-2}. If a SI base unit is not needed for presentation, its exponent equals zero.

In order to convert values given in local units into SI units, normally a multiplication by a conversion factor is required. For example the conversion of acceleration from g into m/s² results in a multiplication by the factor 9.81 (1*g*=9.81*m* s^{-2}). At least in case of temperatures one additionally has to deal with a unit related offset, because the base unit for temperature is Kelvin and its zero point in terms of Centigrade is –273.15 *°C*.

EXAMPLE: **UNIT CONVERSION**

$$\text{Kelvin} = \text{Centigrade} \times 1.0 \frac{\text{K}}{°\text{C}} + 273.15\text{K}$$

$$\text{Kelvin} = \text{Fahrenheit} \times 0.556 \frac{\text{K}}{°\text{F}} + 255.37\text{K}$$

In the first example the factor is 1.0 and the offset is 273.15;
in the second example the factor is 0.556 and the offset is 255.37.

To conclude, the equation for converting a local value "Value" in a "local unit" into a "SI value" in a "SI unit" is as follows:

$$\text{SI}-\text{Value}\left[\prod_{i=1}^{7}\text{SI}-\text{BU}_i^{\text{Expn}_i}\right] = \text{Value}\left[\text{LocalUnit}\right] \times \text{Factor}\left[\frac{\prod_{i=1}^{7}\text{SI}-\text{BU}_i^{\text{Expn}_i}}{\text{LocalUnit}}\right] + \text{Offset}\left[\prod_{i=1}^{7}\text{SI}-\text{BU}_i^{\text{Expn}_i}\right]$$

The following information has to be stored:

- the name of the local unit (an Ascii text)
- all seven signed integer exponents of the SI base units,
- the factor and
- the offset (both as real numbers).

Table of SI units:

Index	Quantity kind	unit name	symbol
1	Length	Meter	m
2	Mass	Kilogramme	kg
3	Time	Second	s
4	Current	Ampere	A
5	Temperature	Kelvin	K
6	Molar Amount	mol	mol
7	Luminous Intensity	Candela	cd

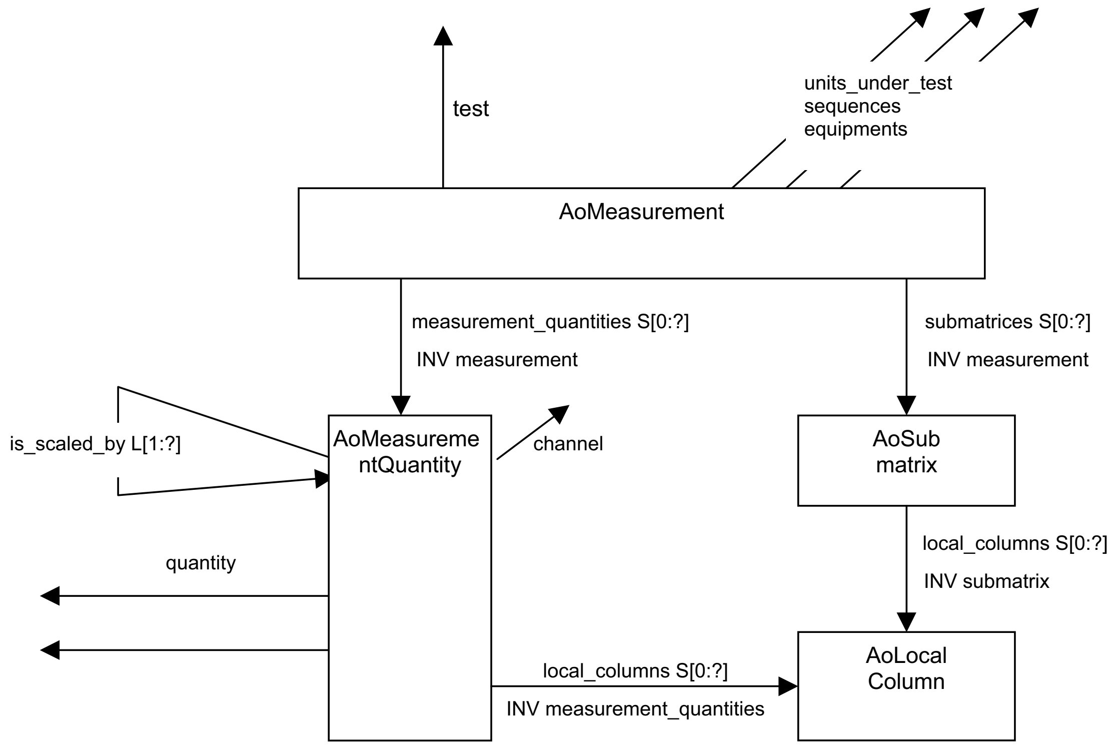
The exponents for the dimensions of all seven SI base units must be stored as attributes of "AoPhysicalDimension". If the corresponding SI base unit does not contribute to the physical dimension, 0 will be stored in that attribute).

Remark on quantities without units and units of dimensionless quantities

"Quantities" with certain "value formats" e.g. Character or Bit do not possess any "units". In contrast to that there exist dimensionless "quantities", which possess "units" with all exponents = 0 (e.g. [-], [%], or others).

4.4 DESCRIPTION OF BASE ELEMENTS FOR MEASUREMENTS

4.4.1 OVERVIEW



Besides these, the base element AoExternalComponent provides a means to store mass data in external files and reference them within the ASAM ODS model.

Please note the distinction in the cardinality specifications:

S[0:?] = set of zero to many, order of elements in the set is irrelevant

L[0:?] = list of zero to many, order is maintained

4.4.2 AOMEASUREMENT (BID = 3)

The following table describes the base attributes of AoMeasurement:

Base Attributes	Meaning
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance, only unique within its predecessor in a hierarchy
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
measurement_begin	DT_DÀTE OPTIONAL Time stamp at measurement begin
measurement_end	DT_DÀTE OPTIONAL Time stamp at measurement end
Relations:	
test	FATHER AoTestAbstract
units_under_test	INFO_REL SET[0:?] OF AoUnitUnderTestAbstract
sequences	INFO_REL SET[0:?] OF AoTestSequenceAbstract
equipments	INFO_REL SET[0:?] OF AoTestEquipmentAbstract
measurement_quantities	CHILD SET[0:?] OF AoMeasurementQuantity
submatrices	CHILD SET[0:?] OF AoSubmatrix

AoMeasurement is the linking point for all data that relate to one test run.

AoMeasurement has references to

- the descriptive data of that test run. This includes the equipment used and its settings, the subject that is being tested and its configuration, and the sequence of steps performed during that test run.
- the result data of the test run. They contain all measurement results, calculation results, status results and also eventually the predefined setpoints. This additionally includes a description of all those quantities that were observed, calculated, or used as setpoint variables.

4.4.3 AOMEASUREMENTQUANTITY (BID = 4)

The following table describes the base attributes of AoMeasurementQuantity:

Base Attributes	Meaning
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance, only unique within its predecessor in a hierarchy
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
datatype	datatype_enum Format in which the data is stored
rank	DT_LONG OPTIONAL Rank of a tensor, number of value dimensions
dimension	DT_LONG LIST[0:?] OPTIONAL Number of values for each rank
type_size	DT_LONG OPTIONAL Length limit of a value, for example the maximum length of a string
interpolation	interpolation_enum OPTIONAL Which type is used when needed, during interpolation
minimum	DT_DOUPLE OPTIONAL Minimum value
maximum	DT_DOUPLE OPTIONAL

	Maximum value
average	DT_DOUPLE OPTIONAL Average
standard_deviation	DT_DOUPLE OPTIONAL Standard deviation
Relations:	
quantity	INFO_TO AoQuantity
unit	INFO_TO AoUnit
local_columns	INFO_FROM SET[0:?] OF AoLocalColumn
channel	INFO_TO AoTestEquipment OR AoTestEquipmentPart OR AoTestDevice OPTIONAL
is_scaled_by	INFO_FROM LIST[1:?] OF AoMeasurementQuantity OPTIONAL Used for scaling the cells. For multidimensional scaling (rank>1) the rightmost index is incremented in the innermost loop
measurement	FATHER AoMeasurement INVERSE FOR measurement_quantities
scales	INFO_TO SET[0:1] OF AoMeasurementQuantitiy INVERSE FOR is_scaled_by

A "measurement quantity" further describes a column of the value matrix. It expresses the use of a "quantity" during "measurement". The specified "measurement quantity" appears only once in one "measurement". The "measurement quantities" may be related to each other, i.e. one quantity may be depending on another one. Particularly for time flow measurements this relation is used between a measurement quantity and the corresponding time quantity.

EXAMPLE: **DEPENDENCIES BETWEEN MEASUREMENT QUANTITIES**

A measurement quantity may depend on the
- torque in a "full load" measurement
- time in a time series measurement during a test cycle

4.4.4 SUBMATRICES AND THE VALUE MATRIX

For a better understanding of the two base elements *AoSubmatrix* and *AoLocalColumn,* the following sections are intended to explain the interdependencies of submatrices, value matrices and local columns.

"Submatrices" are the objects which physically store mass data. Their use is based on technical reasons for storing: While the “large” value matrix has the advantage of transparency and immediate access to the order of measurement points, the submatrices hold the values in a much more compact storage area by not storing “no values” for all positions in the value matrix, in which “measurement quantities” have not been measured at certain “measurement points”.

"Submatrices" are generally homogenous subareas of a value matrix. In this context “homogenous” means that all values exist, while “inhomogenous” matrices may have “holes”, i.e. areas for which no values are provided. The inhomogenous value matrix of the "measurement" may be created out of the corresponding "submatrices" using the specified "link instructions" stored in the "measurement".

There are link instructions for the submatrices:

1. Two submatrices are assigned to each other through the measurement quantity "time" ("time" belongs to both submatrices as a column!). The link instruction is stored and is executed each time when reading data. The instruction implies global "measured points", which are not, however, stored and because of that they do not exist explicitly (e.g. as measured point number).
2. The global "measured point" is stored. The rows of the submatrices are assigned using the global "measured point". The link instruction has therefore not to be executed once again while reading.

Mixed forms of both cases may appear (example wind-tunnel: the place measurement is linked to the real measurement through a link instruction using the quantity "place number". The "submatrices" that build the real measurement are linked through "measured point numbers”).

The API supports both methods: It stores and executes the link instructions as well as generates and stores measured points by the aid of the link instruction.

EXAMPLE:

4.4.4.1 CREATING MEASURED POINTS

With ‘Start_Measured_Point’ a new "measured point number" is generated.
While writing each of the submatrices TM1 and TM2 with ‘Write_Submatrix_Row’ a "measured point number” is stored in every submatrix.

```
Start_Measured_Point
Write_Submatrix_Row(TM1)
Write_Submatrix_Row(TM2)
End_Measured_Point
```

The following example shall explain the use of matrices by displaying them as tables. This should enable tracing of particular values on their way from measurement to submatrices and value matrix.

EXAMPLE:

4.4.4.2 CONSTRUCTING A VALUE MATRIX

Two devices perform measurements independently from each other.
Each of the devices stores the measured values locally, and thus two submatrices are created.
The assignment of measured values is done using the time (each device has its own time measurement, the devices can be synchronized to one point in time).
The two submatrices are mixed with regard to the time:

Submatrix 1		Submatrix 2	
Time	**Quantity_A**	**Time**	**Quantity_B**
0.1	xxx	0.22	yyy
0.2	xxx	0.24	yyy
0.3	xxx	0.26	yyy
0.4	xxx	0.28	yyy
0.5	xxx	0.30	yyy
		0.32	yyy
		0.34	yyy
		0.35	yyy
		0.38	yyy
		0.4	yyy
		0.42	yyy

The resulting value matrix is:

Time	Quantity_A	Quantity_B
0.10	xxx	
0.20	xxx	
0.22		yyy
0.24		yyy
0.26		yyy
0.28		yyy
0.30	xxx	yyy
0.32		yyy
0.34		yyy
0.36		yyy
0.38		yyy
0.40	xxx	yyy
0.42		yyy
0.50	xxx	

The two submatrices are stored separately and linked together afterwards

4.4.5 AoSubmatrix (BID = 38)

The following table describes the base attributes of AoSubmatrix:

Base Attributes	**Meaning**
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance, only unique within its predecessor in a hierarchy
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
number_of_rows	DT_LONG number of „measured points“ in this submatrix
Relations:	
local_columns	CHILD LIST[0:?] OF AoLocalColumn
measurement	FATHER AoMeasurement INVERSE FOR submatrices

4.4.6 AOLOCALCOLUMN (BID = 39)

The following table describes the base attributes of AoLocalColumn:

Base Attributes	Meaning
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance, only unique within its predecessor in a hierarchy
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
global_flag	DT_SHORT A 2-byte integer whose bits have specific meanings
flags	DT_SHORT LIST[1:?] OPTIONAL List of 2-byte integers with bitwise meaning, the length must be the same as the length of “values” and must coincide with the attribute “number_of_rows” in the submatrix.
independent	DT_SHORT 1 – independent 0 – dependent only one local_column per submatrix may be independent
minimum	DT_DOUBLE OPTIONAL Minimum
maximum	DT_DOUBLE OPTIONAL Maximum
sequence_representation	seq_rep_enum Enumeration with possible values: “explicit”, “implicit_linear”, “implicit_constant”, “implicit_saw”, “raw_linear”, “raw_poly-

	nomial", "formula"
generation_parameters	DT_DOUBLE LIST[1:?] OPTIONAL List of parameters necessary to calculate the values for implicit and raw local columns. Omitted for explicit local_columns. Length of list and meaning: [n=position in sequence] [x=sequence to be generated] [y=sequence given in value_sequence] explicit x[n] = y[n] (i.e. no action!) implicit_constant x[n] = a implicit_linear x[n] = a + b * (n-1) implicit_saw x[n] = a + b * ((n-1) mod c) raw_linear x[n] = a + b * y[n] raw_polynomial x[n] = a + b * y[n] + c * y[n]^2 + ... formula x[n] is calculated elsewhere based on y[1]
raw_datatype	datatype_enum OPTIONAL The optional datatype enumeration value that describes the datatype of the raw data; must be one of the enumeration items of datatype_enum.
Relations:	
values	value_sequence OPTIONAL a value_sequence containing a value for each "measured point", the number of values is given in the attribute "number_of_rows" in the submatrix. Omitted for implicit local columns.
external_component	CHILD LIST[1:?] OF AoExternalComponent OPTIONAL
submatrix	FATHER AoSubmatrix INVERSE FOR local_columns
measurement_quantity	INFO_TO AoMeasurementQuantity INVERSE FOR local_columns

A column which is local to a submatrix.

The attribute "sequence_representation" may have one of the following enumeration values; they are defined as "seq_rep_enum":

```
  explicit                          (=0)
  implicit_constant                 (=1)
  implicit_linear                   (=2)
  implicit_saw                      (=3)
  raw_linear                        (=4)
```

```
raw_polynomial                  (=5)
formula                         (=6)
external_component              (=7)
raw_linear_external             (=8)
raw_polynomial_external         (=9)
raw_linear_calibrated           (=10)
raw_linear_calibrated_external  (=11)
```

This enumeration may be extended by ASAM ODS in the future.

The “seq_rep_enum” determines what can be expected to be in the attributes of AoLocalColumn. The following table gives an overview:

seq_rep_enum	**parameters**	**values**	**external_component**
explicit	---	final values	---
implicit	gener.param	gener.param	---
raw	gener.param	raw values	---
ext_comp	---	values from file	ext.comp.description
raw + extern	gener.param	raw values	ext.comp.description

The generation parameters shall contain for:

`explicit`	---
`implicit_constant`	constant value (offset)
`implicit_linear`	start_value+increment
`implicit_saw`	start_value+increment+number_of_values_per_saw
`raw_linear`	offset + factor
`raw_polynomial`	N(order) + coeff0 + coeff1 + ... + coeffN
`external_component`	---
`raw_linear_external`	const.value+gradient (offset+factor)
`raw_polynomial_external`	N(order) + coeff0 + coeff1 + ... + coeffN
`formula`	---
`raw_linear_calibrated`	offset + factor + calibration
`raw_linear_calibrated_external`	offset + factor + calibration

The data shall contain for:

`explicit`	all the data
`implicit_constant`	constant value (offset)
`implicit_linear`	start_value+increment (offset+factor)
`implicit_saw`	start_value+increment+number_of_values_per_saw
`raw_linear`	raw data
`raw_polynomial`	raw data
`external_component`	all the data
`raw_linear_external`	raw data
`raw_polynomial_external`	raw data
`formula`	the formula string
`raw_linear_calibrated`	raw data
`raw_linear_calibrated_external`	raw data

The attribute external_component is required and contains one or more references to AoExternalComponent for:

```
 external_component
 raw_linear_external
 raw_polynomial_external
 raw_linear_calibrated_external
```

and empty for all other choices.

4.4.7 AoExternalComponent (BID = 40)

The following table describes the base attributes of AoExternalComponent:

Base Attributes	Meaning
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance, only unique within its predecessor in a hierarchy
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
ordinal_number	DT_LONG OPTIONAL Ordinal Number
length_in_bytes	DT_LONG Length in Bytes
filename_url	DT_STRING URL of the File Name
value_type	typespec_enum See description below for details on this enumeration.
start_offset	DT_LONG Start Offset
block_size	DT_LONG Block Size
valuesperblock	DT_LONG Values per Block
value_offset	DT_LONG Value Offset
flags_filename_url	DT_STRING OPTIONAL URL of the File Name with Flags
flags_start_offset	DT_LONG

	OPTIONAL Offset where the Flags begin
Relations:	
local_column	FATHER AoLocalColumn INVERSE FOR external_component

The meaning of "typespec_enum" is the enumeration of

```
DT_BOOLEAN,
DT_BYTE,
DT_SHORT,
DT_LONG,
DT_LONGLONG,
IEEEFLOAT4,
IEEEFLOAT8,
DT_SHORT_BEO,
DT_LONG_BEO,
DT_LONGLONG_BEO,
IEEEFLOAT4_BEO,
IEEEFLOAT8_BEO,
DT_STRING,
DT_BYTESTR,
DT_BLOB
```

This enumeration may be extended by ASAM ODS in the future.

4.5 DESCRIPTION OF THE BASE ELEMENTS FOR ADMINISTRATION

The following elements are used to build a tree structured administration for measurements enabling several levels of subtests:

AoTest (derived from AoTestAbstract)

AoSubtest (derived from AoTestAbstract)

4.5.1 AOTEST (BID = 36)

The following table describes the base attributes of AoTest:

Base Attributes	**Meaning**
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
Relations:	
children	CHILD SET[0:?] OF AoSubTest OR SET[0:?] OF AoMeasurement
environment	FATHER SET[0:1] OF AoEnvironment INVERSE FOR tests

4.5.2 AOSUBTEST (BID = 2)

The following table describes the base attributes of AoSubTest:

Base Attributes	**Meaning**
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance, only unique within its predecessor in a hierarchy
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
Relations:	
parent_test	FATHER AoTest OR AoSubTest
children	CHILD SET[0:?] OF AoSubTest OR SET[0:?] OF AoMeasurement

4.6 DESCRIPTION OF THE BASE ELEMENTS FOR DESCRIPTIVE DATA

The following elements are used to build structures for complementary descriptions of the tests:

for the Tested Object

AoUnitUnderTest (derived from AoUnitUnderTestAbstract)

AoUnitUnderTestPart (derived from AoUnitUnderTestAbstract)

for the Test Sequence

AoTestSequence (derived from AoTestSequenceAbstract))

AoTestSequencePart (derived from AoTestSequenceAbstract)

for the Test Equipment

AoTestEquipment (derived from AoTestEquipmentAbstract)

AoTestEquipmentPart (derived from AoTestEquipmentAbstract)

AoTestDevice (derived from AoTestEquipmentPart)

4.6.1 AOUNITUNDERTEST (BID = 21)

The following table describes the base attributes of AoUnitUnderTest:

Base Attributes	Meaning
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
Relations:	
children	CHILD SET[0:?] OF AoUnitUnderTestPart
environment	FATHER SET[0:1] OF AoEnvironment INVERSE FOR uuts
measurement	INFO_REL SET[0:?] OF AoMeasurement INVERSE FOR units_under_test

4.6.2 AOUNITUNDERTESTPART (BID = 22)

The following table describes the base attributes of AoUnitUnderTestPart:

Base Attributes	**Meaning**
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance, only unique within its predecessor in a hierarchy
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
Relations:	
children	CHILD SET[0:?] OF AoUnitUnderTestPart
measurement	INFO_REL SET[0:?] OF AoMeasurement INVERSE FOR units_under_test
parent_unit_under_test	FATHER AoUnitUnderTest INVERSE FOR children May not exist if parent_unit_under_test_part exists.
parent_unit_under_test_part	FATHER AoUnitUnderTestPart INVERSE FOR children May not exist if parent_unit_under_test exists.

4.6.3 AOTESTSEQUENCE (BID = 25)

The following table describes the base attributes of AoTestSequence:

Base Attributes	Meaning
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
Relations:	
children	CHILD SET[0:?] OF AoTestSequencePart
environment	FATHER SET[0:1] OF AoEnvironment INVERSE FOR sequences
measurement	INFO_REL SET[0:?] OF AoMeasurement INVERSE FOR sequences

4.6.4 AoTestSequencePart (BID = 26)

The following table describes the base attributes of AoTestSequencePart:

Base Attributes	Meaning
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance, only unique within its predecessor in a hierarchy
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
Relations:	
children	CHILD SET[0:?] OF AoTestSequence Part
measurement	INFO_REL SET[0:?] OF AoMeasurement INVERSE FOR sequences
parent_sequence	FATHER SET[1:1] OF AoTestSequence INVERSE FOR children May not exist if parent_sequence_part exists.
parent_sequence_part	FATHER SET[1:1] OF AoTestSequencePart INVERSE FOR children May not exist if parent_sequence exists.

4.6.5 AoTestEquipment (BID = 23)

The following table describes the base attributes of AoTestEquipment:

Base Attributes	**Meaning**
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
Relations:	
children	CHILD SET[0:?] OF AoTestequipmentPart OR AoTestDevice The set may contain instances of both base types.
environment	FATHER SET[0:1] OF AoEnvironment INVERSE FOR equipments
measurement	INFO_REL SET[0:?] OF AoMeasurement INVERSE FOR equipments

4.6.6 AoTestEquipmentPart (BID = 24)

The following table describes the base attributes of AoTestEquipmentPart:

Base Attributes	**Meaning**
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance, only unique within its predecessor in a hierarchy
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
Relations:	
children	CHILD SET[0:?] OF AoTestEquipmentPart OR AoTestDevice The set may contain instances of both base types.
measurement	INFO_REL SET[0:?] OF AoMeasurement INVERSE FOR equipments
parent_equipment	FATHER AoTestEquipment INVERSE FOR children May not exist if parent_equipment_part exists.
parent_equipment_part	FATHER AoTestEquipmentPart INVERSE FOR children May not exist if parent_equipment exists.

4.6.7 AOTESTDEVICE (BID = 37)

The following table describes the base attributes of AoTestDevice:

Base Attributes	Meaning
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance, only unique within its predecessor in a hierarchy
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
Relations:	
children	CHILD SET[0:?] OF AoTestDevice
measurement	INFO_REL SET[0:?] OF AoMeasurement INVERSE FOR equipments
parent_equipment	FATHER AoTestEquipment INVERSE FOR children May not exist if parent_equipment_part exists.
parent_equipment_part	FATHER AoTestEquipmentPart OR AoTestDevice INVERSE FOR children May not exist if parent_equipment exists.

4.7 DESCRIPTION OF THE BASE ELEMENTS FOR SECURITY

The following elements are provided for the storage of security data:

AoUser

AoUserGroup

4.7.1 AOUSER (BID = 34)

The following table describes the base attributes of AoUser:

Base Attributes	**Meaning**
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
password	DT_STRING Contains the password
Relations:	
groups	INFO_REL SET[0:?] OF AoUserGroup INVERSE FOR users

4.7.2 AOUSERGROUP (BID = 35)

The following table describes the base attributes of AoUserGroup:

Base Attributes	Meaning
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
superuser_flag	DT_SHORT Flag, which indicates whether being Superuser or not
Relations:	
users	INFO_REL SET[0:?] OF AoUser

4.8 DESCRIPTION OF THE BASE ELEMENTS FOR OTHER DATA

The following elements are provided for the storage of other data:

AoAny

AoLog

AoParameter

AoParameterSet

4.8.1 AOANY (BID = 0)

The following table describes the base attributes of AoAny:

Base Attributes	Meaning
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
Relations:	
parent	FATHER SET[0:1] OF AoAny INVERSE FOR children
children	CHILD SET[0:?] OF AoAny OPTIONAL

If the parent relation does not exist, the combination of name and version must be unique within the ASAM ODS environment.

4.8.2 AOLOG (BID = 43)

The following table describes the base attributes of AoLog:

Base Attributes	**Meaning**
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
date	DT_DATE Date of the Logbook entry
Relations:	
parent	FATHER SET[0:1] OF AoLog INVERSE FOR children
children	CHILD SET[0:?] OF AoLog

4.8.3 PARAMETERS AND PARAMETER SETS

ASAM-ODS already defined several rules and possibilities to store any type of measurement data (e.g. MIME type, implicit channels, independent channels, ...).

Especially the ways for "normal" data like X-Y-Values are very well defined. So it is possible to load and UNDERSTAND data with every ASAM ODS compliant client written by another compliant product before.

But there are lots of other data, e.g. data coming out from an analysis like statistics or other algorithms (FFT, Classifications, ...). ASAM ODS has already reached the goal of accessing any data, but what is with the goal of "UNDERSTANDING" the data written by another client? These data normally have to be described and handled by additional information and implicit knowledge about the semantics, the parameters and special attributes.

There should be a way defined by ASAM ODS how this can be handled by interested companies. They have to be able to handle those kind of data in ASAM ODS by using the rules and possibilities in an ODS-conform way without any contradictions.

The User's goal is clear: Not only X-Y-data should be saved in a product-independent way. Any result of analysis data should be accessible (already done by ASAM ODS) but furthermore understandable from every product knowing about the data type.

Example:

Imagine there are some test data with measurements in an ODS database. The measurement data are time series of some measurement channels (e.g. durability test in the automotive industry).

Especially in this environment it is quite normal to reduce the amount of data by calculating a very special statistical matrix, the so called "Rainflow matrix".

There are several products on the market supporting this algorithm and data but every product in another format/way. Only storing the matrix inside an ODS database would lead to accessibility but not to an understanding between matrices of different products. The same understanding can only be reached by a companion standard based on the ODS definitions.

Therefore, ASAM ODS has to support a uniform way for handling such companion standards and describing any kind of data including the semantics via any parameters.

For this purpose ASAM ODS supports the base elements „AoParameter" and "AoParameterSet".

4.8.3.1 DESCRIPTION OF AOPARAMETER

AoParameter holds one parameter of any ODS-conform type. It has a name (the name of the parameter) and a value (of any type). Base attributes are besides the standard ODS base attributes:

- pvalue: the value of the instance, stored as a string representation.
- parameter_datatype: The data type of pvalue, given as enumeration value out of the possible values of the DataType enumeration. It is used to interpret pvalue correctly.

4.8.3.2 DESCRIPTION OF AOPARAMETERSET

The AoParameterSet is a “grouping object” for parameters. It holds a set of AoParameter referring to it. Base attributes are the standard ODS base attributes.

4.8.3.3 RELATIONS IN THE BASE MODEL

AoParameterSet has a relation to:

- any number of AoParameter (so that it knows which AoParameter are grouped within it)

AoParameter has relations to:

- exactly one AoParameterSet (the one it belongs to)
- an AoUnit that gives a means to calculate the physical value out of the value information stored for the AoParameter

4.8.3.4 USAGE OF PARAMETERS IN APPLICATION MODELS

Parameters are thought to be linked to any other elements and thereby to describe their characteristics more precisely. The base model however does not include a base relation that may be used for this link. (If it did, one would find a relation to nearly all other base elements, which is not useful. In this aspect it behaves similar to AoAny.)

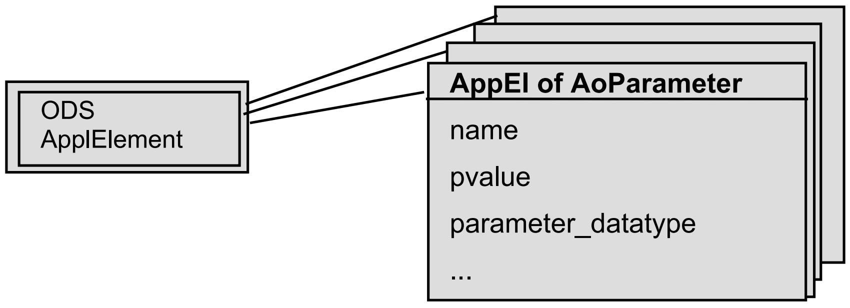
Instead, an application model typically will build application elements of type AoParameter and add application relations to them depending on the information that they carry and for which other application elements this information is of importance.

It is possible to have

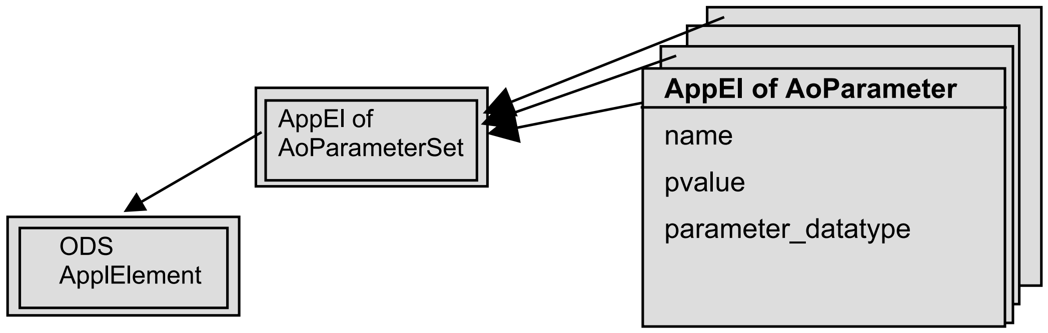
- either an application element of type AoParameterSet (that is referenced by a set of application elements of type AoParameter)
- or an application element of type AoParameter

referring to any other application element.

The following figure shows the situation where an application element of type AoParameter refers to another ODS application element. This is typically used for small numbers of parameters.



The following figure shows the situation where an application element of type AoParameterSet refers to another ODS application element, and where a set of application elements of type AoParameter refer to it:



4.8.3.5 COMPARISON WITH ALTERNATIVE APPROACHES

There are in principle three alternatives to add specific information to application elements:

- One may want to add application attributes that carry the additional information required. In this case the names of the attributes and their data types must be specified at the time the application model is defined.
  Moreover each instance will have to carry the value of that attribute (even in case the same attribute is relevant for a couple of instances).
- One may want to not model the additional information at all within the application model, but to add it by means of instance attributes to all those instances that need the information.
  Again, each instance that needs the information will have to carry the value of that attribute (even in case the same information is relevant for a couple of instances).
- Finally there is the option to define application elements of type AoAny and link them to those elements that required further descriptive information. This is the least specific approach; it does not provide a uniform method of grouping, and attributes that carry the value and the data type and that link to a unit are not provided; they must be added when defining the application model.

By supporting / using the base elements AoParameter and AoParameterSet ASAM ODS can integrate nearly every result type including the descriptive parameters in a flexible but uniform way.

EXAMPLE:

RAINFLOW DATA

The following parameters are necessary for the interpretation of rainflow classifications; they are defined as instances of an application element of type AoParameter for the Rainflow Classification Type.
All these instances must be related to an instance of an an application element of type AoParameterSet.
The column „Flag" shows if the information is mandatory (M) or optional (O); the name is reserved anyway.

Name	Flag	Range/Type	Exception	Remark
Class Minimum	M	-oo …. + oo FLOAT	Minimum>Maximum ➔ Wrong extremes	Minimum of the Classification Range
Class Maximum	M	-oo …. + oo FLOAT	Minimum>Maximum ➔ Wrong extremes	Maximum of the Classification Range
NClasses	M	1...100 SHORT INTEGER	<1 ➔ no valid nClasses	Number of Classes, alternative to BinSize; NClasses=(ClassMaximum – ClassMinimum)/BinSize
BinSize	O	-oo … +oo FLOAT	binSize > (Max–Min) ➔ no valid BinSize	Number of Classes, alternative to NClasses; BinSize=(ClassMaximum –ClassMinimum)/NClasses
Amplitude Suppression	M	1.0 ... nClasses FLOAT	AS < binSize ➔ no valid amplitude suppression	Range of the filter in BinSize (1.0=1 BinSize) Default = 1.0
Symmetric	O	True, False BOOL	-	If set it is a symmetric matrix Default, if not set = True

4.8.4 AOPARAMETER (BID = 44)

The following table describes the base attributes of AoParameter:

Base Attributes	Meaning
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance, only unique within its predecessor in a hierarchy
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
parameter_datatype	datatype_enum
pvalue	DT_STRING The parameter values are always expressed as a string, non-string types are converted accordingly.
Relations:	
unit	AoUnit
parameter_set	FATHER AoParameterSet INVERSE FOR parameters

The parameter_datatype may contain any enumeration element of the base model data type enumeration described in section 2.5.2.

4.8.5 AOPARAMETERSET (BID = 45)

The following table describes the base attributes of AoParameterSet:

Base Attributes	**Meaning**
id	DT_LONGLONG Unique ID for the instances of an application element
name	DT_STRING Name of the instance
description	DT_STRING OPTIONAL Describing text for the instance
version	DT_STRING OPTIONAL Version of the instance
version_date	DT_DATE OPTIONAL Date of the version change
mime_type	DT_STRING OPTIONAL The MIME type of the instance
external_references	DT_EXTERNALREFERENCE LIST[0:?] References to external information
objecttype	DT_LONGLONG OPTIONAL Contains the ID of the application element (subclass or superclass) to which the instance belongs.
Relations:	
parameters	CHILD SET[0:?] OF AoParameter

4.9 FORMAL DESCRIPTION OF THE ASAM ODS BASE MODEL

Using the methodology of ISO 10303 the ASAM ODS base model is defined here. Some explanations are given between the definitions and a graphical representation is added to support comprehension.

4.9.1 WHY STANDARDIZED REPRESENTATION

The task of defining one base model for a large number of different but harmonizing application models requires a description with high demands regarding precision and processing. To avoid inaccuracies in understanding a strictly defined language must be used for description. Using a worldwide standard guarantees a long life-cycle and high reliability of the model description and furthermore offers worldwide available tools to process the description to automate derivations and implementations.

Such internationally standardized description is available in ISO 10303 (“Standard for the Exchange of Product Model Data”, short: STEP). This standard contains a data description language called “Express” accompanied by

- a graphical representation (Express-G),
- prescriptions for physical data storage and access,
- a great variety of data models (basic models called “resources” and derived modes called “application protocols”)

A large number of tools is available, including Express compilers, parsers for both metadata and data, and many applications e.g. in CAD, engineering, or process industry.

For reasons of description precision and international compliance the ASAM ODS base model was described using ISO 10303-11 (Express).

4.9.2 METAMODEL DESCRIPTION IN ISO 10303

This section shall give an introduction to the Express language and the way how entities are defined and represented.

Express is a language for data modeling standardized in ISO 10303 Part 11. In Express the appearance, representation and handling of the data are described in form of a model schema. A model in Express contains the entities (i.e. data types) with their names and attributes, rules, functions and definitions. There are two representations of Express:

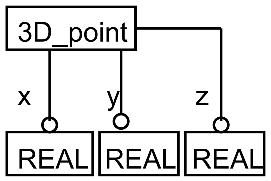
- a complete one in ASCII format (given in the first part of the next section),
- a simplified one in graphical format and therefore called Express-G (given in the second part of the next section).

An entity definition in Express can easily be understood by using an example:

EXAMPLE:

This example shows the definition of a simple 3D point. As can be seen, a 3D_point has three attributes (with names “x”, “y”, “z”) and each of them is of type REAL.

```
ENTITY 3D_point;
  x : REAL;
  y : REAL
  z : REAL;
END_ENTITY;
```

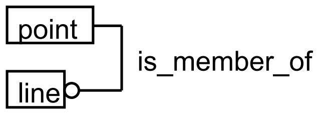


Writing data models in Express requires to give each attribute a name and a type. Such attributes may also be relationships, in which case the type of the attribute is the name of the entity to which the relationship points. Unlike in relational database management systems, in data modeling there is no principle difference between attributes that point to primitive data types (like integer, boolean, string,...) and attributes that point to other entities.

EXAMPLE:

A point may be a member of a line. In that case it is related to an entity "line" which can be expressed by:

```
ENTITY point;
  is_member_of : line;
END_ENTITY;
```



If a relationship is needed in both directions, this can be modeled by explicitly specifying the inverse of an attribute. A definition of an inverse attribute is preceded by the keyword “INVERSE”.

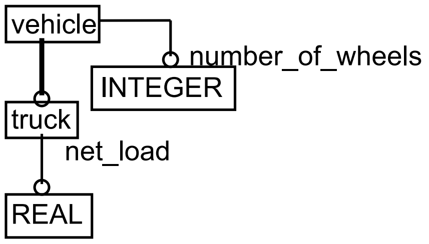
Express allows also to describe derivation of entities, i.e. creating subtypes of an existing entity that inherit all attributes from their supertype and may have further attributes of their own.

In Express-G entity-attribute relationships are represented by thin lines, while supertype-subtype relationships are represented by thick lines.

EXAMPLE:

This example shows the difference between attribute and inheritance relationships. The truck has two attributes: number_of_wheels (inherited from the supertype vehicle) and net load.

```
ENTITY vehicle;
  number_of_wheels : INTEGER;
END_ENTITY;
ENTITY truck
  SUBTYPE OF vehicle;
  net_load : REAL;
END_ENTITY;
```



ASAM users know the process of derivation of application elements from base elements. This is a typical supertype-subtype relationship and is mapped to Express as follows:

- each ASAM ODS base element is represented in Express by an ENTITY.
- each ASAM ODS application element is represented in Express by an ENTITY, too.
- each ENTITY representing an application element is specified in Express as being a subtype of the ENTITY representing the corresponding base element.

Very often the usage of attributes is subject to constraints regarding the allowed values (value range), the types, or the number of items. In Express such restrictions are formulated in rules. Rules are preceded by the keyword “WHERE”.

Sometimes within these rules it is necessary to perform calculations, checks or evaluations. This is usually done in separate functions, often located in a separate section of the Express file.

A data model written in Express is called „schema“. Several tools are available that examine such a schema, check it, and configure a data repository accordingly; such tools are called „Express compilers“.

Writing data models in Express and processing such schemas is part of the methodology of ISO 10303 – the “STEP methodology”.

4.9.3 BASE MODEL REPRESENTATION IN EXPRESS

```
SCHEMA asam28;
(*
ASAM-ODS Base Model                               25 Mar 2004

25.03.2004: AoLocalColumn has been extended by an attribute raw_datatype
24.11.2003: Little adjustments in comments
- writing of DT_EXTERNALREFERENCE and dt_bytestr
- upper case for T_XXX
- STRING as attribute type replaced by T_STRING
20.3.2003 Model 27 for ASAM ODS 4.2 (due in June 2003)
- sequence_representation extended by "calibrated"
13.12.2002 Model 27
- adjust where rules and uniqueness rules
- remove blob sequence, add boolean and longlong sequence
17.10.2002 Model 27 for ASAM ODS 4.2 (preliminary)
- extend seq_rep_enum
- new elements AoNameMap/AoAttributeMap and
   AoParameter/AoParameterSet
- AoEnvironment extended by Application_model_type and rules
- some attributes in AoMeasurement and AoMeasurementQuantity
   become optional
(9.5.2002) Model 26 for ASAM ODS 4.2 (study)
- AoLog added
- Modifications in AoExternal component
  * new attributes ordninal_number and length_in_bytes
  * attribute "blocksize" changed to "block_size"
  * value_type added / value_offset changed in AoExternalComponent
    plus addition of typespec_enum
- AoAny extended with father/child relation
- Data type DT_ENUM added for enumerations
- still to do: rephrase unique rules
(19.9.2001) Extended AoEnvironment + modified Instance Attrs
(03.9.2001) Model 25 with improved relationship types and more
            inverse attributes
(12.7.2001) Model 25 for Discussion
(10.7.2001) For ODS-Version 4.1
- ModelMapper commented out
- independent (flag in AoLocalColumn) is t_short
- superuser_flag (in AoUserGroup) is t_short
- AoEnvironment corrected and old Environment integrated
- instance attributes of all types with units
- integration of attributes of all types with units and flags
- relation types given for reference attributes
(30.5.2001) Corrections for inheritance support:
- Attribute "objecttype" of asam_base_entity with "optional"
```

```
  declarator.
- Rules for Test and Measurement were adjusted to avoid
  contradictions.
(20.02.2001) To be used for ODS-API 4.1
- Derived from models 22 (3.2) and 23 (4.0)
- Proposed entities added: AoEnvironment, AoModelMapper,
   AoExternalComponent
(14.09.00) Base Model version 23 valid for ODS-API 4.0
           (derived from version 21)       ===========
(7.9.2000) Derived version 22 (from 21) only valid for ODS-API 3.2
(13.07.00) Changes in version 21 (from version 20):
- To be used for Version 3.2 and prepared for Version 4.0
- Several attributes marked as "optional", some enumerations cut,
- Several entities marked as "NOT FOR VERSION 3.2" are not to
  be used by API before Version 4.0
- no 64-bit integer, no complex, no boolean numbers
- AoEnvironment inserted provisionally for compatibility
  reasons and for eventual future use.
(15.06.00) Changes in version 20 (from version 19):
- Security added again
- Simple data types more precise
(20.01.00) Changes in version 19 (from version 17):
- Adding MimeTypes and ExternalReference
- No security entities
(17.05.99) Changes in version 18 (from version 17):
- Adding security entities
(03.02.99) Changes in version 17 (from version 16):
- Implicit, raw data and formula data added in AoLocalColumn
- subtypes "implicit_sequence" and "formula_sequence" removed
- division in numeric and textual sequences
(30.10.98) Model 16.

Dr. Helmut J. Helpenstein, National Instruments Engineering
phone   02408 1438 525  or   0049 2408 1438 525
fax     02408 1438 190  or   0049 2408 1438 190
e-mail  helmut.helpenstein@ni.com  or  100135.2174@compuserve.com
*)


ENTITY asam_base_entity
 ABSTRACT SUPERTYPE OF (ONEOF(
   AoEnvironment,
  (*AoModelMapper,*)
   AoTestAbstract,
   AoMeasurement,
   AoMeasurementQuantity,
   AoSubmatrix,
   AoLocalColumn,
   AoExternalComponent,
   AoUnitUnderTestAbstract,
   AoTestEquipmentAbstract,
   AoTestSequenceAbstract,
   AoQuantity,
   AoUnit,
   AoPhysicalDimension,
   AoQuantityGroup,
   AoUnitGroup,
   AoUser,
   AoUserGroup,
   AoLog,
   AoParameter,
   AoParameterSet,
   AoNameMap,
   AoAttributeMap,
   AoAny
  ) );
  name                 : T_STRING;
  id                   : T_LONGLONG;
  version              : OPTIONAL T_STRING;
  description          : OPTIONAL T_STRING;
  version_date         : OPTIONAL T_DATE;
  mime_type            : OPTIONAL T_STRING;
  external_references : LIST [0:?] OF T_EXTERNALREFERENCE;
  instance_attributes : SET [0:?] OF instance_attribute;
  objecttype           : OPTIONAL T_LONGLONG;
                                      (*appl_id for inheritance*)
END_ENTITY;
(*
  This supertype lets all base entities inherit the attributes
  that each of them needs. Being an "abstract" supertype it is
  not to be instantiated.
  name,id: used for identifying any instance of an Asam entity
  The mime types are special strings with the following pattern:
   application/x.asam.ods.AOAOAO.XXXXXX
  with AOAOAO = name of an AsamOds base element
  and  XXXXXX = user defined name of a specialisation
  external_reference usually points to files, but may also point
  to remote documents (e.g. available via WWW)
  The objecttype supports implementations that handle IDs of
  application elements for convenience reasons.
*)



(* ---------------------------------- *)
(*    types and auxiliary entites     *)
(* ---------------------------------- *)


TYPE T_EXTERNALREFERENCE = LIST[3:3] OF T_STRING;
  (* 1st string: description
     2nd string: mime type
     3rd string: location (asam path or URL)
   *)
END_TYPE;


ENTITY AoNameMap                    (* BID=46 *)
 SUBTYPE OF (asam_base_entity);
  entity_name       : T_STRING;
  alias_names       : LIST[0:?] OF T_STRING;
  attribute_mapping : SET [0:?] OF AoAttributeMap;     (*CHILD*)
INVERSE
  environment       : AoEnvironment FOR entity_mapping;(*INFO_TO*)
UNIQUE
  UR1: SELF\asam_base_entity.id;
WHERE
  WR1: SIZEOF(SELF.alias_names) =
       SIZEOF(SELF.environment.meaning_of_aliases);
END_ENTITY;
(*
  For the entity name any number of alias names (e.g. for
  language versions) may be specified.
  The list allows different language version switched by
  the application software.
  The length of the list shall be equal to the list
  "meaning_of_aliases" in AoEnvironment (Where Rule 1).
  For each entity a name mapping may be given;
  For each attribute of the entity an attribute mapping may be given.
*)

ENTITY AoAttributeMap               (* BID=47 *)
 SUBTYPE OF (asam_base_entity);
  attribute_name    : T_STRING;
  alias_names       : LIST[0:?] OF T_STRING;
INVERSE
  name_mapping      : AoNameMap FOR attribute_mapping; (*FATHER*)
UNIQUE
  UR1: SELF\asam_base_entity.id;
WHERE
  WR1: SIZEOF(SELF.alias_names) =
       SIZEOF(SELF.name_mapping.alias_names);
END_ENTITY;
(*
  Specifies any number of alias names for the attribute
  "attribute_name".
  The length of the list of alias names shall be equal to the
  length of that list in AoNameMap (Where Rule 1).
*)
```

```
ENTITY AoParameter       (* BID=44 *)
 SUBTYPE OF (asam_base_entity);
  unit    : AoUnit;
  parameter_datatype : datatype_enum;
  pvalue  : T_STRING;(*always string, other types are converted*)
INVERSE
  parameter_set : AoParameterSet for parameters; (*FATHER*)
UNIQUE
  UR1: SELF\asam_base_entity.id;
END_ENTITY;

ENTITY AoParameterSet    (* BID=45 *)
 SUBTYPE OF (asam_base_entity);
  parameters : SET [0:?] OF AoParameter;         (*CHILD*)
UNIQUE
  UR1: SELF\asam_base_entity.id;
END_ENTITY;



TYPE instance_attribute = SELECT
   (  byte_inst_attribute,
     short_inst_attribute,
      long_inst_attribute,
     float_inst_attribute,
    double_inst_attribute,
   complex_inst_attribute,
  dcomplex_inst_attribute,
    string_inst_attribute,
      time_inst_attribute );
END_TYPE;
(*
  Instance attributes allow giving one or more values with a
  specific meaning to any instance.
  The meaning shall be expressed in their attribute "name".
  The value(s) shall be put into the "values" attribute of
  one of the subtypes according to the data type.
  The entities for the instance attributes are defined in the
  family of attribute_object.
*)


ENTITY attribute_object
 ABSTRACT SUPERTYPE OF (ONEOF(
      byte_abs_attribute,
     short_abs_attribute,
      long_abs_attribute,
     float_abs_attribute,
    double_abs_attribute,
  complex_abs_attribute,
 dcomplex_abs_attribute,
    string_abs_attribute,
      time_abs_attribute ) );
  unit  : OPTIONAL AoUnit;
  flag  : OPTIONAL T_SHORT;
END_ENTITY;

(*
  For each datatype we provide three instanciable subtypes
  of attribute_object:
  xxxx_attribute        with  1.unit,2.flag,3.svalue
  xxxx_seq_attribute    with  1.unit,2.flag,3.values
  xxxx_inst_attribute   with  1.unit,2.flag,3.values,4.name
    (xxxx = float/long/string/...)

  These attribute objects can be used for
  a) instance attributes
  b) application attributes -> then their data type is not
      primitive (e.g. plain float or list of float) but it
      is an object with
      - in case of instance attributes: a name
      - a value (or values)
      - an optional unit
      - an optional flag (16 bits)
  An attribute "quantity" was not provided.

  An application attribute may always be
    - a simple (primitive) attribute: a value or list of values
    - an attribute object like described above.
  This must be valid also for those application attributes
  which are inherited from base attributes.
  Therefore a select type containing all options is created for
  each datatype, its name is:
  xxxx_attr_select   (xxxx = float/long/string/...)
*)

ENTITY byte_abs_attribute
 ABSTRACT SUPERTYPE OF (ONEOF(
    byte_seq_attribute,
    byte_attribute))
 SUBTYPE OF (attribute_object);
END_ENTITY;

ENTITY byte_attribute
 SUPERTYPE OF (byte_inst_attribute)
 SUBTYPE OF (byte_abs_attribute);
    svalue : T_BYTE; (* "svalue" represents a single value *)
                     (* ("value" would be a reserved word) *)
END_ENTITY;
```

```
ENTITY byte_seq_attribute
 SUBTYPE OF (byte_abs_attribute);
   values : LIST[1:?] OF T_BYTE;
END_ENTITY;

ENTITY byte_inst_attribute
 SUBTYPE OF (byte_attribute);
   name  : T_STRING;
END_ENTITY;


ENTITY short_abs_attribute
 ABSTRACT SUPERTYPE OF (ONEOF(
   short_seq_attribute,
   short_attribute))
 SUBTYPE OF (attribute_object);
END_ENTITY;

ENTITY short_attribute
 SUPERTYPE OF (short_inst_attribute)
 SUBTYPE OF (short_abs_attribute);
   svalue : T_SHORT;
END_ENTITY;

ENTITY short_seq_attribute
 SUBTYPE OF (short_abs_attribute);
   values : LIST[1:?] OF T_SHORT;
END_ENTITY;

ENTITY short_inst_attribute
 SUBTYPE OF (short_attribute);
   name  : T_STRING;
END_ENTITY;


ENTITY long_abs_attribute
 ABSTRACT SUPERTYPE OF (ONEOF(
   long_seq_attribute,
   long_attribute))
 SUBTYPE OF (attribute_object);
END_ENTITY;

ENTITY long_attribute
 SUPERTYPE OF (long_inst_attribute)
 SUBTYPE OF (long_abs_attribute);
   svalue : T_LONG;
END_ENTITY;

ENTITY long_seq_attribute
```

```
 SUBTYPE OF (long_abs_attribute);
   values : LIST[1:?] OF T_LONG;
END_ENTITY;

ENTITY long_inst_attribute
 SUBTYPE OF (long_attribute);
   name  : T_STRING;
END_ENTITY;


ENTITY float_abs_attribute
 ABSTRACT SUPERTYPE OF (ONEOF(
   float_seq_attribute,
   float_attribute))
 SUBTYPE OF (attribute_object);
END_ENTITY;

ENTITY float_attribute
 SUPERTYPE OF (float_inst_attribute)
 SUBTYPE OF (float_abs_attribute);
   svalue : T_FLOAT;
END_ENTITY;

ENTITY float_seq_attribute
 SUBTYPE OF (float_abs_attribute);
   values : LIST[1:?] OF T_FLOAT;
END_ENTITY;

ENTITY float_inst_attribute
 SUBTYPE OF (float_attribute);
   name  : T_STRING;
END_ENTITY;


ENTITY double_abs_attribute
 ABSTRACT SUPERTYPE OF (ONEOF(
   double_seq_attribute,
   double_attribute))
 SUBTYPE OF (attribute_object);
END_ENTITY;

ENTITY double_attribute
 SUPERTYPE OF (double_inst_attribute)
 SUBTYPE OF (double_abs_attribute);
   svalue : T_DOUBLE;
END_ENTITY;

ENTITY double_seq_attribute
 SUBTYPE OF (double_abs_attribute);
   values : LIST[1:?] OF T_DOUBLE;
```

```
END_ENTITY;

ENTITY double_inst_attribute
 SUBTYPE OF (double_attribute);
   name  : T_STRING;
END_ENTITY;


ENTITY string_abs_attribute
 ABSTRACT SUPERTYPE OF (ONEOF(
   string_seq_attribute,
   string_attribute))
 SUBTYPE OF (attribute_object);
END_ENTITY;

ENTITY string_attribute
 SUPERTYPE OF (string_inst_attribute)
 SUBTYPE OF (string_abs_attribute);
   svalue : T_STRING;
END_ENTITY;

ENTITY string_seq_attribute
 SUBTYPE OF (string_abs_attribute);
   values : LIST[1:?] OF T_STRING;
END_ENTITY;

ENTITY string_inst_attribute
 SUBTYPE OF (string_attribute);
   name  : T_STRING;
END_ENTITY;


ENTITY time_abs_attribute
 ABSTRACT SUPERTYPE OF (ONEOF(
   time_seq_attribute,
   time_attribute))
 SUBTYPE OF (attribute_object);
END_ENTITY;

ENTITY time_attribute
 SUPERTYPE OF (time_inst_attribute)
 SUBTYPE OF (time_abs_attribute);
   svalue : T_DATE;
END_ENTITY;

ENTITY time_seq_attribute
 SUBTYPE OF (time_abs_attribute);
   values : LIST[1:?] OF T_DATE;
END_ENTITY;
```

```
ENTITY time_inst_attribute
 SUBTYPE OF (time_attribute);
   name  : T_STRING;
END_ENTITY;


ENTITY complex_abs_attribute
 ABSTRACT SUPERTYPE OF (ONEOF(
   complex_seq_attribute,
   complex_attribute))
 SUBTYPE OF (attribute_object);
END_ENTITY;

ENTITY complex_attribute
 SUPERTYPE OF (complex_inst_attribute)
 SUBTYPE OF (complex_abs_attribute);
   svalue : LIST[2:2] OF T_FLOAT;
END_ENTITY;

ENTITY complex_seq_attribute
 SUBTYPE OF (complex_abs_attribute);
   values : LIST[2:?] OF T_FLOAT;
WHERE
  WR1: SIZEOF(values) MOD 2 = 0;
END_ENTITY;

ENTITY complex_inst_attribute
 SUBTYPE OF (complex_attribute);
   name  : T_STRING;
END_ENTITY;


ENTITY dcomplex_abs_attribute
 ABSTRACT SUPERTYPE OF (ONEOF(
   dcomplex_seq_attribute,
   dcomplex_attribute))
 SUBTYPE OF (attribute_object);
END_ENTITY;

ENTITY dcomplex_attribute
 SUPERTYPE OF (dcomplex_inst_attribute)
 SUBTYPE OF (dcomplex_abs_attribute);
   svalue : LIST[2:2] OF T_DOUBLE;
END_ENTITY;

ENTITY dcomplex_seq_attribute
 SUBTYPE OF (dcomplex_abs_attribute);
   values : LIST[2:?] OF T_DOUBLE;
WHERE
  WR1: SIZEOF(values) MOD 2 = 0;
```

```
END_ENTITY;

ENTITY dcomplex_inst_attribute
 SUBTYPE OF (dcomplex_attribute);
   name  : T_STRING;
END_ENTITY;


(* THE types for most attributes: *)

TYPE byte_attr_select = SELECT
  ( T_BYTE,
    byte_sequence,
    byte_attribute,
    byte_seq_attribute );
END_TYPE;

TYPE short_attr_select = SELECT
  ( T_SHORT,
    short_sequence,
    short_attribute,
    short_seq_attribute );
END_TYPE;

TYPE long_attr_select = SELECT
  ( T_LONG,
    long_sequence,
    long_attribute,
    long_seq_attribute );
END_TYPE;

TYPE float_attr_select = SELECT
  ( T_FLOAT,
    float_sequence,
    float_attribute,
    float_seq_attribute );
END_TYPE;

TYPE double_attr_select = SELECT
  ( T_DOUBLE,
    double_sequence,
    double_attribute,
    double_seq_attribute );
END_TYPE;

TYPE complex_attr_select = SELECT
  ( T_COMPLEX,
    complex_sequence,
    complex_attribute,
    complex_seq_attribute );
```

```
END_TYPE;

TYPE dcomplex_attr_select = SELECT
  ( T_DCOMPLEX,
    dcomplex_sequence,
    dcomplex_attribute,
    dcomplex_seq_attribute );
END_TYPE;

TYPE time_attr_select = SELECT
  ( T_DATE,
    time_sequence,
    time_attribute,
    time_seq_attribute );
END_TYPE;

TYPE string_attr_select = SELECT
  ( T_STRING,
    string_sequence,
    string_attribute,
    string_seq_attribute );
END_TYPE;



TYPE T_BOOLEAN = BOOLEAN;
  (*  1 bit *)
END_TYPE;

TYPE T_BYTE = INTEGER;
  (*  8 bit *)
END_TYPE;

TYPE T_SHORT = INTEGER;
  (* 16 bit *)
END_TYPE;

TYPE T_LONG = INTEGER;
  (* 32 bit *)
END_TYPE;

TYPE T_LONGLONG = INTEGER;
  (* 64 bit *)
END_TYPE;

(*TYPE T_ID = t_longlong;
   * 64 bit *
END_TYPE;*)

TYPE T_FLOAT = REAL;
```

```
  (* 32 bit *)
END_TYPE;

TYPE T_DOUBLE = REAL;
  (* 64 bit *)
END_TYPE;

TYPE T_COMPLEX = LIST [2:2] OF REAL;
  (* 32 bit *)
END_TYPE;

TYPE T_DCOMPLEX = LIST [2:2] OF REAL;
  (* 64 bit *)
END_TYPE;

TYPE T_STRING = STRING;
  (* UTF8 string terminated by 0 *)
END_TYPE;

TYPE T_DATE = STRING;
(* meaning: YYYYMMDDhhmmsslllcccnnn....
   year month day hour minute second millisec microsec nanosec ...*)
END_TYPE;

TYPE T_BYTESTR = STRING;
  (* sequence of bytes *)
END_TYPE;

TYPE T_BLOB = STRING;
  (* 1.header, 2. sequence of bytes *)
END_TYPE;

TYPE datatype_enum = ENUMERATION OF (
  DT_UNKNOWN,    (* 0 , also for "reserved" *)
  DT_STRING,     (* 1 *)
  DT_SHORT,      (* 2 *)
  DT_FLOAT,      (* 3 *)
  DT_BOOLEAN,    (* 4 *)
  DT_BYTE,       (* 5 *)
  DT_LONG,       (* 6 *)
  DT_DOUBLE,     (* 7 *)
  DT_LONGLONG,   (* 8 *)
  DT_ID,         (* 9 , not used in attributes *)
  DT_DATE,       (* 10 *)
  DT_BYTESTR,    (* 11 *)
  DT_BLOB,       (* 12 *)
  DT_COMPLEX,    (* 13 *)
  DT_DCOMPLEX,   (* 14 *)
  DT_EXTERNALREFERENCE, (* 28 *)
  DT_ENUM);      (* 30 *)
```

```
END_TYPE;




(* -------------------------------------- *)
(*     Begin of ASAM-ODS base elements     *)
(* -------------------------------------- *)



ENTITY AoEnvironment  (* BID=1 *) (* previously: BID=42 *)
 SUBTYPE OF (asam_base_entity);               (*relation_types: *)
  tests      : SET [0:?] OF AoTest;           (*CHILD*)
  uuts       : SET [0:?] OF AoUnitUnderTest;  (*CHILD*)
  equipments : SET [0:?] OF AoTestEquipment;  (*CHILD*)
  sequences  : SET [0:?] OF AoTestSequence;   (*CHILD*)
  entity_mapping     : SET [0:?] OF AoNameMap;(*INFO_FROM*)
  meaning_of_aliases : LIST [0:?] OF T_STRING;
  max_test_level            : OPTIONAL T_LONG;
  base_model_version        : OPTIONAL T_STRING;
                             (* schema name, in this case 'asam27' *)
  application_model_type    : OPTIONAL T_STRING;
                     (* may contain many names, comma-separated *)
  application_model_version : OPTIONAL T_STRING;
                             (* any operator-supplied name *)
UNIQUE
  UR1: SELF\asam_base_entity.id;
WHERE
  WR1: unique_name_and_version(tests);
  WR2: unique_name_and_version(uuts);
  WR3: unique_name_and_version(equipments);
  WR4: unique_name_and_version(sequences);
    (* There shall not be 2 members of the sets with
       identical name and version. *)
END_ENTITY;
(*
  There should be only one instance of environment.

  The server name should be stored in the attribute
  "name" which is inherited from asam_base_entity.

  Only instances of the top entities of the following 4
  hierarchical trees are collected in the sets:
  - AoTest
  - AoUnitUnderTest
  - AoTestEquipment
  - AoTestSequence
  .
  In entity_mapping all the maps are to be collected.
  The set contains one AoNameMap per entitiy.
  An alias_index in the application software allows to use the
  appropriate alias_name from the given lists.
  In this way language versions can be switched easily.
  For each position in the lists of alias names a string should
  be provided in "meaning_of_aliases"; this allows for
  name mapping to different languages.
  The attribute "max_test_level" shows the number of levels in
  the test hierarchy.
*)

(*
ENTITY AoModelMapper  (  BID=41  )
 SUBTYPE OF (asam_base_entity);
  ruletext         : T_STRING;
UNIQUE
  UR1: SELF\asam_base_entity.id;
END_ENTITY;
*)

(*
  Application elements are built by deriving subtypes from the
  base entities given in this schema.
  For the management of measurements and tests the following
  rules have to be followed:

1.In the whole model only one entity of type AoTest (or
  a derived subtype) may be instantiated. This one is the main
  test (e.g. an application element called "Testseries").

2.Assuming no entities of type AoSubTest (or its subtype)
  are used, then the attribute "children" of the AoTestAbstract
  element "Testseries" is always of the type "Measurement",
  i.e. children of an instance of "Testseries" are all
  instances of "Measurement". It is not allowed to have
  children of type "Testseries".
  The Test-Measurement-Hierachy then has the form
      Testseries –o Measurement

3.Assuming there is one application element derived from
  AoSubTest (e.g. called "Maintest"),
  the attribute "children" of "Testseries" is of type "Maintest",
  and the attribute "parent_test" of "Maintest" is of type
  "Testseries".
  The attribute "children" of "Maintest" is of type "Measurement".
  The Test-Measurement-Hierarchie then has the form
      Testseries –o Maintest –o Measurement

4.Assuming more than one application element have been
  derived from AoSubTest (e.g. "Maintest", "Subtest" and
  "Subsubtest"), then they have to be build strictly as a
  chain:
      Testseries -o Maintest -o Subtest -o Subsubtest
                                                  |
                                                  o
                                              Measurement

  In this chain the uppermost application element of
  AoSubTest (here "Maintest") has a "parent_test" attribute
  of type "Testseries" (the AoTest element), and the
  lowermost application element of AoSubTest (here
  "Subsubtest") has a "children" attribute of type
  "Measurement".
  The other "parent_test" and "children" attributes of the
  inner AoSubTest application elements always have exactly
  the type of the corresponding application elements in the
  chain (e.g. the "parent_test" attribute of "Subtest" is of
  type "Maintest", and the "children" attribute of "Subtest"
  is of type "Subsubtest").
  So the rule says: It is not allowed to have a "parent_test"
  relation to an instance of another application element than
  the one defined in the chain as the parent test.
  An analogous rule is valid for the "children" attribute.
*)

ENTITY AoTestAbstract
 ABSTRACT SUPERTYPE OF (ONEOF(
  AoTest,
  AoSubTest ) )
 SUBTYPE OF (asam_base_entity);
  children: SET [0:?] OF test_measurement_select;  (*CHILD*)
UNIQUE
  UR1: SELF\asam_base_entity.id;
WHERE
  WR1: unique_name_and_version(children);
    (* There shall not be 2 members of this set with
       identical name and version. *)
END_ENTITY;

ENTITY AoTest  (* BID=36 *)
 SUBTYPE OF (AoTestAbstract);
INVERSE
 environment : SET [0:1] OF AoEnvironment for tests; (*FATHER*)
UNIQUE
  UR1: SELF\asam_base_entity.name,
       SELF\asam_base_entity.version;
END_ENTITY;

ENTITY AoSubTest (* BID=2 *)
 SUBTYPE OF (AoTestAbstract);
  parent_test : AoTestAbstract;   (*FATHER*)
END_ENTITY;
```

```
TYPE test_measurement_select = SELECT
 ( AoSubTest,
   AoMeasurement );
END_TYPE;
(*
  only the AoTestAbstract in the bottommost level shall have
  an AoMeasurement, in higher levels it shall be an
  AoSubTest.
*)


ENTITY AoMeasurement (* BID=3 *)
 SUBTYPE OF (asam_base_entity);
  test                  : AoTestAbstract;           (*FATHER*)
  measurement_quantities: SET [0:?] OF AoMeasurementQuantity;
                                                    (*CHILD*)
  submatrices           : SET [0:?] OF AoSubmatrix; (*CHILD*)
  units_under_test      : SET [0:?] OF AoUnitUnderTestAbstract;
                                                    (*INFO_REL*)
  sequences             : SET [0:?] OF AoTestSequenceAbstract;
                                                    (*INFO_REL*)
  equipments            : SET [0:?] OF AoTestEquipmentAbstract;
                                                    (*INFO_REL*)
  measurement_begin     : OPTIONAL T_DATE;
  measurement_end       : OPTIONAL T_DATE;
UNIQUE
  UR1: SELF\asam_base_entity.id;
WHERE
  WR1 : unique_name_and_version(measurement_quantities);
  WR2 : unique_name_and_version(submatrices);
END_ENTITY;


ENTITY AoMeasurementQuantity (* BID=4 *)
 SUBTYPE OF (asam_base_entity);
  local_columns  : SET [0:?] OF AoLocalColumn;       (*INFO_FROM*)
  quantity       : AoQuantity;                       (*INFO_TO*)
  unit           : AoUnit;                           (*INFO_TO*)
  channel        : OPTIONAL AoTestEquipmentAbstract; (*INFO_TO*)
  is_scaled_by   : OPTIONAL LIST [1:?] OF AoMeasurementQuantity;
                   (* used for scaling the cells !
                      For multidimensional scaling (rank>1)
                      the rightmost index is incremented in the
                      innermost loop *)              (*INFO_FROM*)
  datatype       : datatype_enum;
  rank           : OPTIONAL T_LONG; (*of one cell*)
  dimension      : OPTIONAL LIST [0:?] OF T_LONG; (*of one cell*)
  type_size      : OPTIONAL T_LONG; (*of a scalar in byte*)
  interpolation  : OPTIONAL interpolation_enum;
  minimum        : OPTIONAL T_DOUBLE;
  maximum        : OPTIONAL T_DOUBLE;
  average        : OPTIONAL T_DOUBLE;
  standard_deviation: OPTIONAL T_DOUBLE;
INVERSE
  measurement    : AoMeasurement for measurement_quantities;
                                                    (*FATHER*)
  scales         : SET [0:1] OF AoMeasurementQuantity for
                                      is_scaled_by; (*INFO_TO*)

UNIQUE
  UR1: SELF\asam_base_entity.id;
WHERE
  WR1: SIZEOF(SELF.dimension) = SELF.rank;
    (* The attribute dimension contains so many integers as
       rank (the number of dimensions of a cell) specifies. *)
  WR2: (NOT EXISTS(SELF.is_scaled_by)) XOR
       (SIZEOF(SELF.is_scaled_by)=SELF.rank);
    (* If a scaling exists, then the number of scaling vectors
       is given by rank *)
(*
  WR3: unique_name_and_version(local_columns);
  This rule is commented out, because the attribute "local_columns"
  describes an INFO relation (not FATHER/CHILD).
*)
END_ENTITY;

TYPE interpolation_enum = ENUMERATION OF (
  no_interpolation,
  linear_interpolation,
  application_specific );
END_TYPE;


(*
  state is no longer used.
  It has been replaced by measurement_quantity, its name
  describes the desired state and the values in its
  local columns may be of type boolean.
  For logging events the type may also be string.
*)


ENTITY AoSubmatrix (* BID = 38 *)
 SUBTYPE OF (asam_base_entity);
  number_of_rows : T_LONG;
  local_columns  : LIST [0:?] OF AoLocalColumn;        (*CHILD*)
INVERSE
  measurement    : AoMeasurement for submatrices;     (*FATHER*)
UNIQUE
```

```
   UR1: SELF\asam_base_entity.id;
 WHERE
 (*WR1: NOT EXISTS (SELF\asam_base_entity.version);*)
   WR2: unique_name_and_version(local_columns);
   (* Remark: The function compares also instances
              that have no version. *)
 END_ENTITY;



 (*
    Value representation:
    local columns and sequences, possibly including generation
 *)

 ENTITY AoExternalComponent (* BID=40 *)
  SUBTYPE OF (asam_base_entity);
   ordinal_number     : OPTIONAL T_LONG;
   length_in_bytes    : T_LONG;
   filename_url       : T_STRING;
   value_type         : typespec_enum;
   start_offset       : T_LONG;
   block_size         : T_LONG;
   valuesperblock     : T_LONG;
 (*value_offsets      : LIST [1:?] OF T_LONG; (old) *)
   value_offset       : T_LONG;
   flags_filename_url : OPTIONAL T_STRING;
   flags_start_offset : OPTIONAL T_LONG;
 INVERSE
   local_column       : AoLocalColumn for external_component;
                                                        (*FATHER*)
 UNIQUE
   UR1: SELF\asam_base_entity.id;
 END_ENTITY;

 TYPE typespec_enum = ENUMERATION OF (
    dt_boolean,
    dt_byte,
    dt_short,
    dt_long,
    dt_longlong,
    ieeefloat4,
    ieeefloat8,
    dt_short_beo,
    dt_long_beo,
    dt_longlong_beo,
    ieeefloat4_beo,
    ieeefloat8_beo,
    dt_string,
    dt_bytestr,
    dt_blob );
```

```
END_TYPE;


TYPE seq_rep_enum = ENUMERATION OF (
   explicit,
   implicit_constant,
   implicit_linear,
   implicit_saw,
   raw_linear,
   raw_polynomial,
   formula,
   external_component,
   raw_linear_external,
   raw_polynomial_external,
   raw_linear_calibrated, (*extension March 2003*)
   raw_linear_calibrated_external
   (* extendible!! *) );
END_TYPE;

(*
  The seq_rep_enum determines what can be expected to be in
  the attributes of AoLocalColumn. The following table gives
  an overview:
  attribute generation_                      external_
seq_rep     parameters        values         component
   _enum
explicit       - - -       final values       - - -
implicit    gener.param    gener.param        - - -
raw         gener.param    raw values         - - -
ext_comp       - - -       values from file  ext.comp.description
raw+extern  gener.param    raw values        ext.comp.description

  The generation parameters shall contain for
   explicit                    ---
   implicit_constant           constant value (offset)
   implicit_linear             start_value+increment
   implicit_saw                start_value+increment+number_of_values_per_saw
   raw_linear                  offset + factor
   raw_polynomial              N(order) + coeff0 + coeff1 + ... + coeffN
   external_component          ---
   raw_linear_external         const.value+gradient (offset+factor)
   raw_polynomial_external     N(order) + coeff0 + coeff1 + ... + coeffN
   formula                     ---
   raw_linear_calibrated       offset + factor + calibration
   raw_linear_calibrated_external  offset + factor + calibration

  The data shall contain for
   explicit                    all the data
   implicit_constant           constant value (offset)
   implicit_linear             start_value+increment (offset+factor)
```

```
    implicit_saw                start_value+increment+number_of_values_per_saw
    raw_linear                  raw data
    raw_polynomial              raw data
    external_component          all the data
    raw_linear_external         raw data
    raw_polynomial_external     raw data
    formula                     the formula string
    raw_linear_calibrated       raw data
    raw_linear_calibrated_external  raw data

   The external_component is given for
    external_component
    raw_linear_external
    raw_polynomial_external
    raw_linear_calibrated_external
   and empty for all other choices.

 *)

 ENTITY AoLocalColumn (* BID = 39 *)
  SUBTYPE OF (asam_base_entity);
   flags                    : OPTIONAL LIST [1:?] OF T_SHORT;
   global_flag              : T_SHORT;
   independent              : T_SHORT;
   minimum                  : OPTIONAL T_DOUBLE;
   maximum                  : OPTIONAL T_DOUBLE;
   sequence_representation : seq_rep_enum;
   generation_parameters   : OPTIONAL LIST [1:?] OF T_DOUBLE;
   raw_datatype             : OPTIONAL datatype_enum;
   values                   : OPTIONAL value_sequence;
   external_component      : OPTIONAL LIST [1:?] OF
                                       AoExternalComponent;(*CHILD*)
 INVERSE
   submatrix   : AoSubmatrix for local_columns;          (*FATHER*)
   measurement_quantity : AoMeasurementQuantity for local_columns;
                                                         (*INFO_TO*)
 UNIQUE
   UR1: SELF\asam_base_entity.id;
 WHERE
 (*WR1: NOT EXISTS (SELF\asam_base_entity.version);*)
   WR2: (NOT EXISTS(SELF.flags)) XOR
        (SIZEOF(SELF.flags)=submatrix.number_of_rows);
   WR3: (SELF.sequence_representation = implicit_constant)
          XOR                                (*implicit sequences*)
        (SELF.sequence_representation = implicit_linear)
          XOR
        (SELF.sequence_representation = implicit_saw)
          XOR
        (SELF.sequence_representation = formula)
          XOR
        (sequence_length(SELF.values) =      (*explicit sequences*)
         local_column_list_length(           (*      and raw data*)
           SELF.submatrix.number_of_rows,
           SELF.measurement_quantity.rank,
           SELF.measurement_quantity.dimension) );
 (*
    Explanation of WR3:
    In the attribute "values" there is a value_sequence,
    which has in turn an attribute "values" which is a list.
    This where rule is to ensure the right length of that
    list. For this we need two information items:
    1. from AoSubmatrix given in the (inv) attribute
       "submatrix":
       its attribute "number_of_rows" (an integer);
    2. from AoMeasurementQuantity given in the (inv) attribute
       measurement_quantity:
       2a. its attribute "rank" (an integer)
       2b. its attribute "dimension" (a list of integer)
    Now the required list length can be calculated, the algorithm
    is contained in the function "local_column_list_length":
      list_length = number_of_rows;
      FOR i=1...rank (step 1)     [nothing happens if rank=0]
        list_length = list_length * dimension[i-1];
      END_FOR;
    The required list length is then compared to the actual
    list length provided by function "sequence_length", which
    is also called here. In case of implicit/formula sequences
    the function shall not be evaluated.
 *)
   WR4: (  (  (SELF.sequence_representation = explicit)
          XOR (SELF.sequence_representation = formula) )
        AND
        (      (EXISTS (SELF.values)) AND
          (NOT (EXISTS (SELF.generation_parameters))) ) )
     XOR
        (  (  (SELF.sequence_representation = implicit_constant)
          XOR (SELF.sequence_representation = implicit_linear)
          XOR (SELF.sequence_representation = implicit_saw) )
        AND
        (     EXISTS (SELF.generation_parameters)) )

     XOR
        (  (  (SELF.sequence_representation = raw_linear)
          XOR (SELF.sequence_representation = raw_polynomial)
          XOR (SELF.sequence_representation = raw_linear_calibrated) )
        AND
        (      (EXISTS (SELF.values)) AND
          (     EXISTS (SELF.generation_parameters)) ) )
     XOR
        (  (  (SELF.sequence_representation = external_component)
          XOR (SELF.sequence_representation = raw_linear_external)
          XOR (SELF.sequence_representation = raw_linear_calibrated_external)
          XOR (SELF.sequence_representation = raw_polynomial_external) )
        AND
        (     EXISTS (SELF.external_component)) ) ;

 (*
    For explicit and raw data the attribute "values" is used;
    for implicit and raw data the attribute "generation_parameters"
      is used;
    for formula data the attribute "values" has a string_sequence
      (preferably of length 1) containing the formula.
    for data on external components the attribute "external_component"
      is used, others may be used according to the type.
 *)
   WR5: (  (  (SELF.sequence_representation = implicit_constant)
          XOR (SELF.sequence_representation = implicit_linear)
          XOR (SELF.sequence_representation = implicit_saw)
          XOR (SELF.sequence_representation = raw_linear)
          XOR (SELF.sequence_representation = raw_linear_calibrated)
          XOR (SELF.sequence_representation = raw_polynomial)
          XOR (SELF.sequence_representation = raw_linear_external)
          XOR (SELF.sequence_representation = raw_linear_calibrated_external)
          XOR (SELF.sequence_representation = raw_polynomial_external) )
        AND
        ( 'numeric_sequence' IN TYPEOF (SELF.values) ) )
     XOR
        (      SELF.sequence_representation = explicit)
     XOR
        (     (SELF.sequence_representation = formula )
        AND
        ( 'string_sequence' IN TYPEOF (SELF.values) ) );
 (*
    For explicit data the type of the sequence my be any;
    for implicit and raw data the sequence must be numeric;
    for formula the sequence must be of type string.
 *)
 END_ENTITY;


 ENTITY value_sequence
  ABSTRACT SUPERTYPE OF (ONEOF(
   numeric_sequence,
   textual_sequence) );
 END_ENTITY;

 ENTITY textual_sequence
  ABSTRACT SUPERTYPE OF (ONEOF(
   bytestr_sequence,
 (*blob_sequence,  (removed 12.12.2002)*)
```

```
  string_sequence) )
 SUBTYPE OF (value_sequence);
END_ENTITY;

ENTITY numeric_sequence
 ABSTRACT SUPERTYPE OF (ONEOF(
  boolean_sequence,
  byte_sequence,
  short_sequence,
  long_sequence,
  longlong_sequence,
  float_sequence,
  double_sequence,
  complex_sequence,
  dcomplex_sequence,
  time_sequence ) )
 SUBTYPE OF (value_sequence);
END_ENTITY;

ENTITY boolean_sequence
 SUBTYPE OF (numeric_sequence);
  values : LIST [1:?] OF T_BOOLEAN;
END_ENTITY;

ENTITY byte_sequence
 SUBTYPE OF (numeric_sequence);
  values : LIST [1:?] OF T_BYTE;
END_ENTITY;

ENTITY short_sequence
 SUBTYPE OF (numeric_sequence);
  values : LIST [1:?] OF T_SHORT;
END_ENTITY;

ENTITY long_sequence
 SUBTYPE OF (numeric_sequence);
  values : LIST [1:?] OF T_LONG;
END_ENTITY;

ENTITY longlong_sequence
 SUBTYPE OF (numeric_sequence);
  values : LIST [1:?] OF T_LONGLONG;
END_ENTITY;

ENTITY float_sequence
 SUBTYPE OF (numeric_sequence);
  values : LIST [1:?] OF T_FLOAT;
END_ENTITY;

ENTITY double_sequence
```

```
 SUBTYPE OF (numeric_sequence);
  values : LIST [1:?] OF T_DOUBLE;
END_ENTITY;

ENTITY complex_sequence
 SUBTYPE OF (numeric_sequence);
  values : LIST [2:?] OF T_FLOAT;
(* pairs of real and imaginary part are stored consecutively *)
WHERE
  WR1: SIZEOF(values) MOD 2 = 0;
END_ENTITY;

ENTITY dcomplex_sequence
 SUBTYPE OF (numeric_sequence);
  values : LIST [2:?] OF T_DOUBLE;
(* pairs of real and imaginary part are stored consecutively *)
WHERE
  WR1: SIZEOF(values) MOD 2 = 0;
END_ENTITY;

ENTITY time_sequence
 (* Derive no further subtypes! *)
 SUBTYPE OF (numeric_sequence);
  values : LIST [1:?] OF T_DATE;
END_ENTITY;

ENTITY string_sequence
 (* Derive no further subtypes! *)
 SUBTYPE OF (textual_sequence);
  values : LIST [1:?] OF T_STRING;
END_ENTITY;

ENTITY bytestr_sequence
 (* Derive no further subtypes! *)
 SUBTYPE OF (textual_sequence);
  values : LIST [1:?] OF T_BYTESTR;
END_ENTITY;


(*
  The 3 hierarchical entities
    AoUnitUnderTestAbstract,
    AoTestSequenceAbstract,
    AoTestEquipmentAbstract
  are constructed in the same way. The top
  level is always given by a separate entitiy, an arbitrary
  number of other levels may follow. Hierarchical relation on
  instance level is ensured by references in both directions.
*)
```

```
ENTITY AoUnitUnderTestAbstract
 ABSTRACT SUPERTYPE OF (ONEOF(
  AoUnitUnderTest,
  AoUnitUnderTestPart ) )
 SUBTYPE OF (asam_base_entity);
  children    : SET [0:?] OF AoUnitUnderTestPart;    (*CHILD*)
INVERSE
  measurement : SET[0:?] OF AoMeasurement for units_under_test;
                                                    (*INFO_REL*)
UNIQUE
  UR1: SELF\asam_base_entity.id;
WHERE
  WR1: unique_name_and_version(children);
  (*WR2: EXISTS(version_date);*)
END_ENTITY;

ENTITY AoUnitUnderTest (* BID=21 *)
 SUBTYPE OF (AoUnitUnderTestAbstract);
INVERSE
 environment : SET [0:1] OF AoEnvironment for uuts; (*FATHER*)
UNIQUE
  UR1: SELF\asam_base_entity.name,
       SELF\asam_base_entity.version;
END_ENTITY;

ENTITY AoUnitUnderTestPart (* BID=22 *)
 SUBTYPE OF (AoUnitUnderTestAbstract);
INVERSE
  parent_unit_under_test : AoUnitUnderTest for children;(*FATHER*)
  parent_unit_under_test_part : AoUnitUnderTestPart
                                        for children; (*FATHER*)
WHERE
  WR1: (NOT EXISTS (parent_unit_under_test)) XOR
       (NOT EXISTS (parent_unit_under_test_part));
END_ENTITY;


ENTITY AoTestSequenceAbstract
 ABSTRACT SUPERTYPE OF (ONEOF(
  AoTestSequence,
  AoTestSequencePart ) )
 SUBTYPE OF (asam_base_entity);
  children     : SET [0:?] OF AoTestSequencePart; (*CHILD*)
INVERSE
  measurement  : SET[0:?] OF AoMeasurement for sequences;
                                                  (*INFO_REL*)
UNIQUE
  UR1: SELF\asam_base_entity.id;
WHERE
  WR1: unique_name_and_version(children);
```

```
   (*WR2: EXISTS(version_date);*)
END_ENTITY;

ENTITY AoTestSequence (* BID=25 *)
 SUBTYPE OF (AoTestSequenceAbstract);
INVERSE
 environment : SET [0:1] OF AoEnvironment for sequences;(*FATHER*)
UNIQUE
  UR1: SELF\asam_base_entity.name,
       SELF\asam_base_entity.version;
END_ENTITY;

ENTITY AoTestSequencePart (* BID=26 *)
 SUBTYPE OF (AoTestSequenceAbstract);
INVERSE
  parent_sequence  : AoTestSequence for children; (*FATHER*)

  parent_sequence_part  : AoTestSequencePart
                                   for children; (*FATHER*)
WHERE
  WR1: (NOT EXISTS (parent_sequence)) XOR
       (NOT EXISTS (parent_sequence_part));
END_ENTITY;


ENTITY AoTestEquipmentAbstract
 ABSTRACT SUPERTYPE OF (ONEOF(
  AoTestEquipment,
  AoTestEquipmentPart ) )
 SUBTYPE OF (asam_base_entity);
  children    : SET [0:?] OF AoTestEquipmentPart; (*CHILD*)
INVERSE
  measurement : SET[0:?] OF AoMeasurement for equipments;
                                                 (*INFO_REL*)
UNIQUE
  UR1: SELF\asam_base_entity.id;
WHERE
  WR1: unique_name_and_version(children);
   (*WR2: EXISTS(version_date);*)
END_ENTITY;

ENTITY AoTestEquipment (* BID=23 *)
 SUBTYPE OF (AoTestEquipmentAbstract);
INVERSE
 environment : SET [0:1] OF AoEnvironment for equipments;
                                                   (*FATHER*)
UNIQUE
  UR1: SELF\asam_base_entity.name,
       SELF\asam_base_entity.version;
END_ENTITY;
```

```
ENTITY AoTestEquipmentPart (* BID=24 *)
 SUPERTYPE OF (ONEOF(
  AoTestDevice ) )
 SUBTYPE OF (AoTestEquipmentAbstract);
INVERSE
  parent_equipment : AoTestEquipment for children;   (*FATHER*)

  parent_equipment_part : AoTestEquipmentPart
                                     for children;   (*FATHER*)
WHERE
  WR1: (NOT EXISTS (parent_equipment)) XOR
       (NOT EXISTS (parent_equipment_part));
   (* This formulation of inverse attribute ensures that one
      parent_equipment is always required. It is either of type
      AoTestEquipment or of type AoTestEquipmentPart. The first
      of the two has a relation to AoEnvironment.*)
END_ENTITY;

(*
  The following entity AoTestDevice shall be the link for ASAM-G.
  Further subtypes (application elements) can be derived from it
  in any number of levels.
*)
ENTITY AoTestDevice (* BID=37 *)
 SUBTYPE OF (AoTestEquipmentPart);
WHERE
  WR1: SIZEOF (QUERY(child <* SELF\AoTestEquipmentPart.children |
              (NOT ('AoTestDevice' IN TYPEOF (child)) ) )) = 0;
       (* The children of AoTestDevice must have type
          AoTestDevice (and not AoTestEquipment...) *)
END_ENTITY;



ENTITY AoQuantity (* BID=11 *)
 SUBTYPE OF (asam_base_entity);
  successors        : SET [0:?] OF AoQuantity;  (*CHILD*)
  quantity_classification : OPTIONAL quantity_class_enum;
                           (*default value="measured"*)
   (* The following 6 attributes are MANDATORY since Version 4.0 *)
  default_unit      : AoUnit;                   (*INFO_TO*)
  default_rank      : T_LONG;
  default_dimension : LIST [0:?] OF T_LONG;
  default_datatype  : datatype_enum;
  default_type_size : T_LONG; (* for strings/bytestreams: maximum length *)
  default_mq_name   : T_STRING;
INVERSE
  predecessor : SET [0:1] OF AoQuantity FOR successors; (*FATHER*)
  groups      : SET [0:?] OF AoQuantityGroup for quantities;
```

```
                                                   (*INFO_REL*)
  measurement_quantities: SET [0:?] OF AoMeasurementQuantity for
                                        quantity; (*INFO_FROM*)
UNIQUE
  UR1: SELF\asam_base_entity.id;
  UR2: SELF\asam_base_entity.name;
WHERE
  WR1: SIZEOF(SELF.default_dimension) = SELF.default_rank;
(*
    DO WE STILL NEED THE FOLLOWING RULES ?
  WR2: unique_name_and_version(successors);
  WR3: unique_name_and_version(groups);
*)
END_ENTITY;

TYPE quantity_class_enum = ENUMERATION OF (
   measured,
   state);
END_TYPE;


ENTITY AoUnit (* BID=13 *)
 SUBTYPE OF (asam_base_entity);
  phys_dimension : AoPhysicalDimension;           (*INFO_TO*)
  factor         : T_DOUBLE;
  offset         : T_DOUBLE;
INVERSE
  groups     : SET [0:?] OF AoUnitGroup for units; (*INFO_REL*)
  quantities : SET [0:?] OF AoQuantity for default_unit;
                                                 (*INFO_FROM*)
UNIQUE
  UR1: SELF\asam_base_entity.id;
  UR2: SELF\asam_base_entity.name,phys_dimension;
END_ENTITY;
(*
  factor is the (value using SI unit) when
                (value using this unit) = 1
  offset is the (value using SI unit) when
                (value using this unit) = 0
  :
  (value using this unit) multiplied by
                (factor) = (value using SI unit)
  (value using this unit) plus
                (offset) = (value using SI unit)
  (value using this unit) multiplied by (factor)
           plus (offset) = (value using SI unit)
  :
  Examples:
  (4200) 1/min = (4200 * factor) 1/s     [factor=0.016667]
               = (70) 1/s
  (20) deg.C = (20 + offset) K           [offset=273.15]
             = (293.15) K
  (68) deg.F = (68 * factor + offset) K  [factor=0.55555]
             = (293,15) K                [offset=255.372]
  (200) mm = (200 * factor) m  = 0.2 m   [factor=0.001]
*)


ENTITY AoPhysicalDimension (* BID=15 *)
 SUBTYPE OF (asam_base_entity);
  length_exp             : T_LONG;
  mass_exp               : T_LONG;
  time_exp               : T_LONG;
  current_exp            : T_LONG;
  temperature_exp        : T_LONG;
  molar_amount_exp       : T_LONG;
  luminous_intensity_exp : T_LONG;
  length_exp_den             : OPTIONAL T_LONG;
  mass_exp_den               : OPTIONAL T_LONG;
  time_exp_den               : OPTIONAL T_LONG;
  current_exp_den            : OPTIONAL T_LONG;
  temperature_exp_den        : OPTIONAL T_LONG;
  molar_amount_exp_den       : OPTIONAL T_LONG;
  luminous_intensity_exp_den : OPTIONAL T_LONG;
INVERSE
  units   : SET[0:?] OF AoUnit FOR phys_dimension; (*INFO_FROM*)
UNIQUE
  UR1: SELF\asam_base_entity.id;
  UR2: SELF\asam_base_entity.name;
WHERE
  (* ensure default_value=1 *)
  WR1: EXISTS(length_exp_den) OR (length_exp_den=1);
  WR2: EXISTS(mass_exp_den) OR (mass_exp_den=1);
  WR3: EXISTS(time_exp_den) OR (time_exp_den=1);
  WR4: EXISTS(current_exp_den) OR (current_exp_den=1);
  WR5: EXISTS(temperature_exp_den) OR (temperature_exp_den=1);
  WR6: EXISTS(molar_amount_exp_den) OR (molar_amount_exp_den=1);
  WR7: EXISTS(luminous_intensity_exp_den) OR
       (luminous_intensity_exp_den=1);
  (* ensure denominators not equal zero *)
  WR8 : length_exp_den<>0;
  WR9 : mass_exp_den<>0;
  WR10: time_exp_den<>0;
  WR11: current_exp_den<>0;
  WR12: temperature_exp_den<>0;
  WR13: molar_amount_exp_den<>0;
  WR14: luminous_intensity_exp_den<>0;
END_ENTITY;
```

```
ENTITY AoQuantityGroup (* BID=12 *)
 SUBTYPE OF (asam_base_entity);
  quantities : SET [0:?] OF AoQuantity;  (*INFO_REL*)
UNIQUE
  UR1: SELF\asam_base_entity.id;
  UR2: SELF\asam_base_entity.name;
END_ENTITY;

ENTITY AoUnitGroup (* BID=14 *)
 SUBTYPE OF (asam_base_entity);
  units : SET [0:?] OF AoUnit;           (*INFO_REL*)
UNIQUE
  UR1: SELF\asam_base_entity.id;
  UR2: SELF\asam_base_entity.name;
END_ENTITY;


ENTITY AoLog (* BID=43 *)
 SUBTYPE OF (asam_base_entity);
  date     : T_DATE;
  children : SET [0:?] OF AoLog;                      (*CHILD*)
INVERSE
  parent   : SET [0:1] OF AoLog for children;          (*FATHER*)
UNIQUE
  UR1: SELF\asam_base_entity.id;
  UR2: SELF\asam_base_entity.name,date;
END_ENTITY;


ENTITY AoAny (* BID=0 *)
 SUBTYPE OF (asam_base_entity);
(* Allowed to build a hierachy within the AoAny. *)
 children : OPTIONAL SET [0:?] OF AoAny;               (*CHILD*)
INVERSE
 parent   : SET [0:1] OF AoAny for children;           (*FATHER*)
UNIQUE
  UR1: SELF\asam_base_entity.id;
  UR2: SELF\asam_base_entity.name,
       SELF\asam_base_entity.version,
       parent;
       (* if parent does not exists then unique (name,version) *)
WHERE
  WR1: unique_name_and_version(children);
END_ENTITY;
(*
  Also AoAny should not be instantiated, instead entities should
  be derived from it.
*)
```

```
(*
  SECURITY SECTION
*)

ENTITY AoUser  (* BID=34 *)
 SUBTYPE OF (asam_base_entity);
  password       : T_STRING;
INVERSE
  groups   : SET [0:?] Of AoUserGroup for users; (*INFO_REL*)
UNIQUE
  UR1: SELF\asam_base_entity.id;
  UR2: SELF\asam_base_entity.name;
END_ENTITY;

ENTITY AoUserGroup  (* BID=35 *)
 SUBTYPE OF (asam_base_entity);
  users    : SET [0:?] Of AoUser;                (*INFO_REL*)
  superuser_flag : T_SHORT;
UNIQUE
  UR1: SELF\asam_base_entity.id;
  UR2: SELF\asam_base_entity.name,
       SELF\asam_base_entity.version;
END_ENTITY;


ENTITY ACL     (* not asam_base_entities !! *)
 ABSTRACT SUPERTYPE OF (ONEOF(
   ACLI,
   ACLA,
   ACLTemplate
  ) );
  users             : AoUserGroup;               (*INFO_TO*)
  appl_element_name : T_STRING;
  rights            : T_SHORT;
END_ENTITY;

ENTITY ACLI
 SUBTYPE OF (ACL);
  instance_id       : T_LONGLONG;
END_ENTITY;

ENTITY ACLA
 SUBTYPE OF (ACL);
  attribute_name    : T_STRING; (* if empty: Whole application *)
END_ENTITY;                     (* element with all attributes *)

ENTITY ACLTemplate
 SUBTYPE OF (ACL);
```

```
    instance_id       : T_LONGLONG;
    ref_appl_elem_name: T_STRING;
  (* The table SVCTPLI contains:
    USERGROUPID  Reference to a UserGroup instance
    AID          Application element id
    RIGHTS       Rights value (bitmasked)
    IID          Application instance id
    REFAID       Referencing application element
   *)
  END_ENTITY;

  ENTITY SecurityLevel;
    appl_element_name : T_STRING;
    level             : T_SHORT;
   (*
     bit  decimal   ACL on       Protection on
      0      1      appl elem    all instances
      1      2      inst elem    instances individually
      2      4      elem attrs   attributes individually
     Combination by adding the decimal values.
   *)
  END_ENTITY;

  ENTITY InitialRightsAt;
  (* This is to be contained in table SVCATTR, column ACLREF *)
    appl_element_name : T_STRING;
    attribute_name    : T_STRING;
  END_ENTITY;


  (*
    ---------------------- FUNCTIONS --------------------------
  *)

  (*
    The following function is called from AoLocalColumn. It is
    used to determine the actual size of its value sequences.
  *)

  FUNCTION sequence_length(
      seq : value_sequence
      ): INTEGER;
    LOCAL
      list_length : INTEGER;
    END_LOCAL;
    list_length := 0;
    IF 'boolean_sequence' IN TYPEOF (seq) THEN
       list_length := SIZEOF (seq\boolean_sequence.values);
      END_IF;
    IF 'byte_sequence' IN TYPEOF (seq) THEN
```

```
     list_length := SIZEOF (seq\byte_sequence.values);
    END_IF;
  IF 'short_sequence' IN TYPEOF (seq) THEN
     list_length := SIZEOF (seq\short_sequence.values);
    END_IF;
  IF 'long_sequence' IN TYPEOF (seq) THEN
     list_length := SIZEOF (seq\long_sequence.values);
    END_IF;
  IF 'longlong_sequence' IN TYPEOF (seq) THEN
     list_length := SIZEOF (seq\longlong_sequence.values);
    END_IF;
  IF 'float_sequence' IN TYPEOF (seq) THEN
     list_length := SIZEOF (seq\float_sequence.values);
    END_IF;
  IF 'double_sequence' IN TYPEOF (seq) THEN
     list_length := SIZEOF (seq\double_sequence.values);
    END_IF;
  IF 'complex_sequence' IN TYPEOF (seq) THEN
     list_length := SIZEOF (seq\complex_sequence.values) / 2;
    END_IF;
  IF 'dcomplex_sequence' IN TYPEOF (seq) THEN
     list_length := SIZEOF (seq\dcomplex_sequence.values) / 2;
    END_IF;
  IF 'time_sequence' IN TYPEOF (seq) THEN
     list_length := SIZEOF (seq\time_sequence.values);
    END_IF;
  IF 'string_sequence' IN TYPEOF (seq) THEN
     list_length := SIZEOF (seq\string_sequence.values);
    END_IF;
  IF 'bytestr_sequence' IN TYPEOF (seq) THEN
     list_length := SIZEOF (seq\bytestr_sequence.values);
    END_IF;
  RETURN(list_length);
END_FUNCTION;

(*
  The following function is called from AoLocalColumn. It is
  used to determine the required size of its value sequences.
*)

FUNCTION local_column_list_length(
    number_of_rows : INTEGER;
    rank           : INTEGER;
    dimension      : LIST OF INTEGER
    ): INTEGER;
  LOCAL
    list_length : INTEGER;
    i           : INTEGER;
    temp        : INTEGER;
  END_LOCAL;
  list_length := number_of_rows;
  REPEAT i := 1 TO rank BY 1;  (*nothing happens if rank=0*)
    temp        := list_length * dimension[i-1];
    list_length := temp;
  END_REPEAT;
  RETURN(list_length);
END_FUNCTION;



(*
  The following function is called if the members of a set
  should be distinct in name+version. Since "version" is
  optional, the function also works without it, then it
  checks only that "name" is different.
  The function is here given in Express. It is understood
  that an implementation in a normal programming language
  will look different. But the action of the function should
  be the same:
  - take the set passed in the argument "instances"
  - look if any two members of the set have the same name
     * if NO return TRUE
     * if yes look if those members differ at least in
       the version:
       if YES return TRUE  otherwise  return FALSE
  It is planned to call this function only in cases where
  the uniqueness of set members matters, i.e. when building
  hierarchical trees.
*)
FUNCTION unique_name_and_version (
    instances : AGGREGATE OF asam_base_entity
    ): BOOLEAN;
  LOCAL
    set_size   : INTEGER;
    uniqueness : BOOLEAN;
    version1   : STRING;
    version2   : STRING;
    inst1      : asam_base_entity;
    inst2      : asam_base_entity;
  END_LOCAL;
  uniqueness := TRUE;
  set_size := SIZEOF(instances);
  REPEAT i := 1 TO set_size BY 1;
     version1 := ''; (*modified 13.12.2002*)
     inst1 := instances[i];
     IF EXISTS(inst1.version) THEN
       version1 := inst1.version;
     END_IF;
     REPEAT j := i+1 TO set_size BY 1;
        inst2 := instances[j];
        IF inst1.name = inst2.name THEN
```

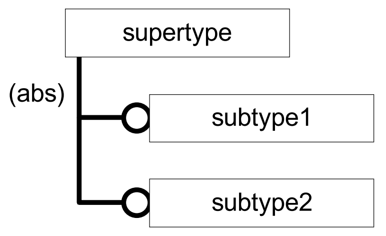
```
           version2 := ''; (*modified 13.12.2002*)
           IF EXISTS(inst2.version) THEN
             version2 := inst2.version;
           END_IF;
           IF version1 = version2 THEN
              uniqueness := FALSE;
              RETURN(uniqueness);
           END_IF;
        END_IF;
     END_REPEAT;
  END_REPEAT;
  RETURN(uniqueness);
END_FUNCTION;

END_SCHEMA;
```

4.9.4 BASE MODEL REPRESENTATION IN EXPRESS-G

The diagrams on the following pages show graphical representations of the Express model given in the previous section, i.e. the graphics are derived from the master model in Express. In case of any difference between the two representations the textual Express model shall prevail. Please note that rules and functions cannot be included in the diagrams, even if they are important for the usage of the model.

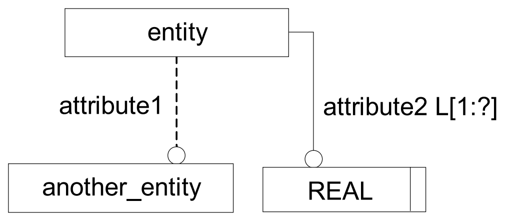
The following diagram (“Explanation of Express-G”) gives you a syntactical overview of Express-G notations. The next diagrams (“Environment” to ”Parameter, Log, External Component”) show a particular section of the ODS base model each. Partly the contents of the diagrams overlap.

Explanation of Express-G

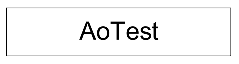


supertype subtype relationship,
if marked with "(abs)" then
supertype is not to be instantiated.

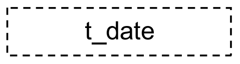
All relationships have a direction that is
indicated with the ball at the end of the line.



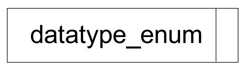
attribute relationship,
solid line = mandatory attribute
dashed line = optional attribute
lines are marked with attribute name
L[a:b] = list with at least a and at most b members
S[a:b] = set with at least a and at most b members
INV means inverse attribute (i.e. relationship in
opposite direction)



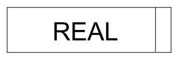
an entity



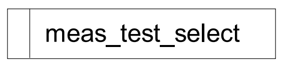
a defined type



an enumeration type (with a predefined set of items)

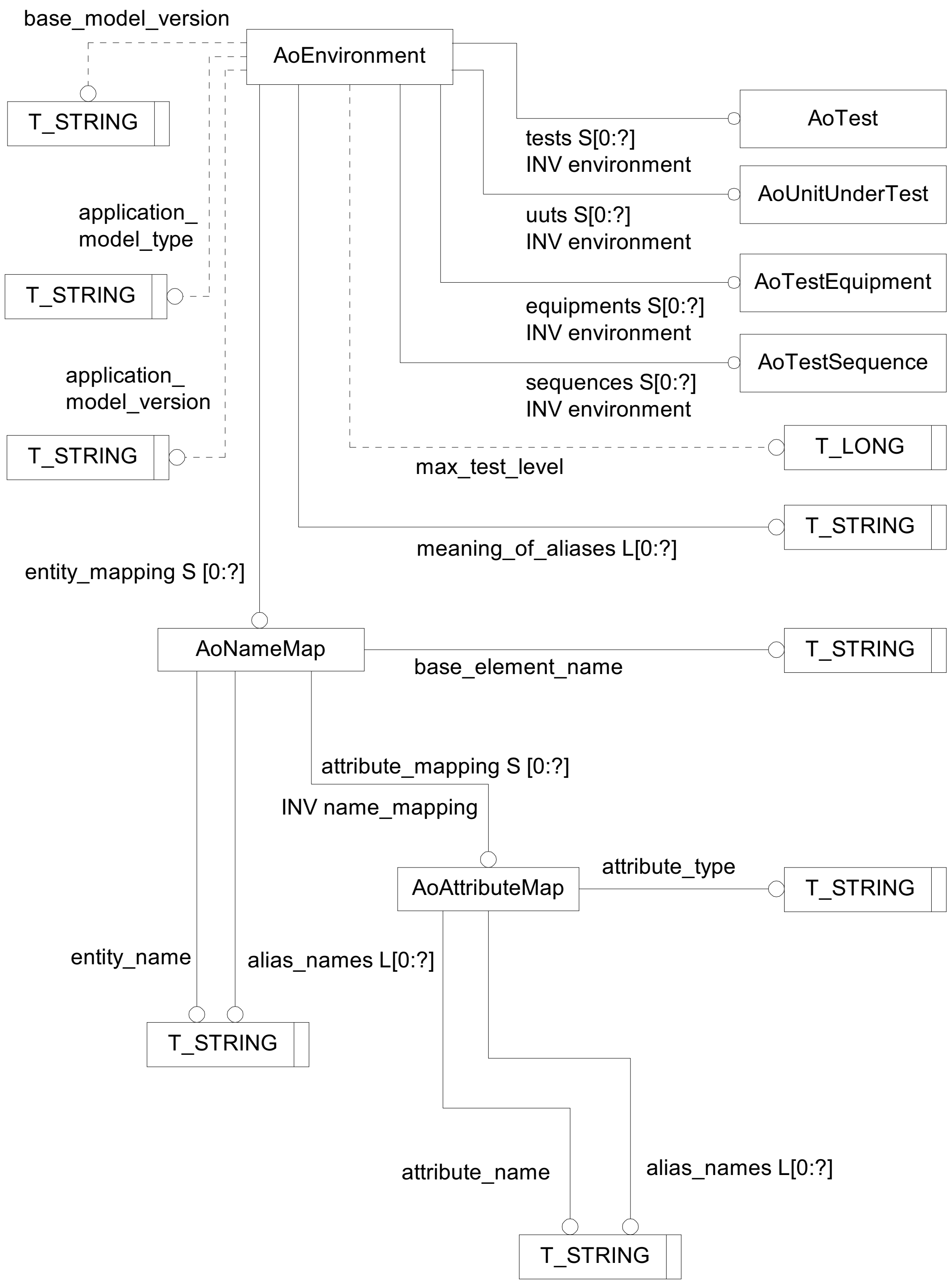


a primitive type

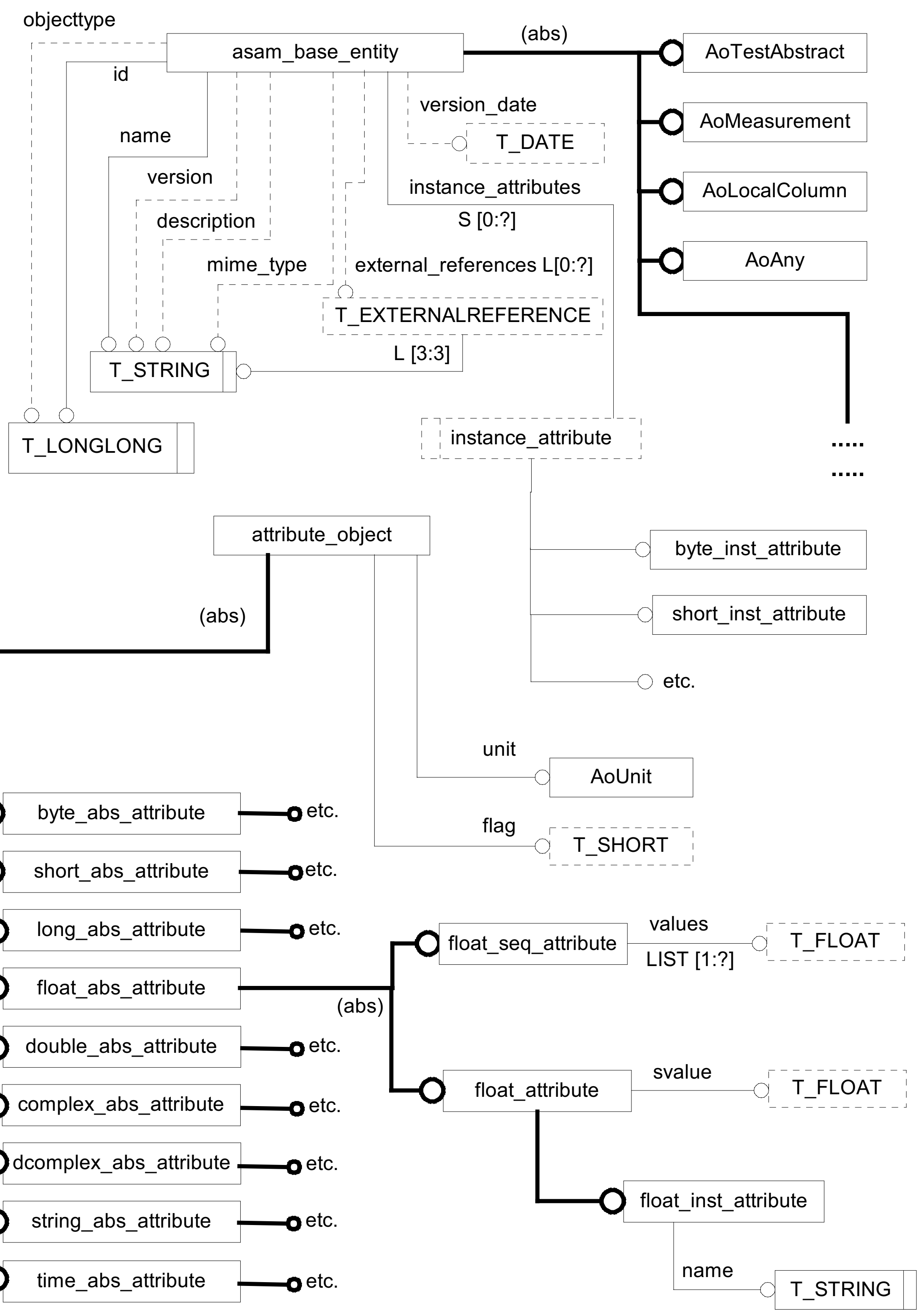


a select type (e.g. an alternative)

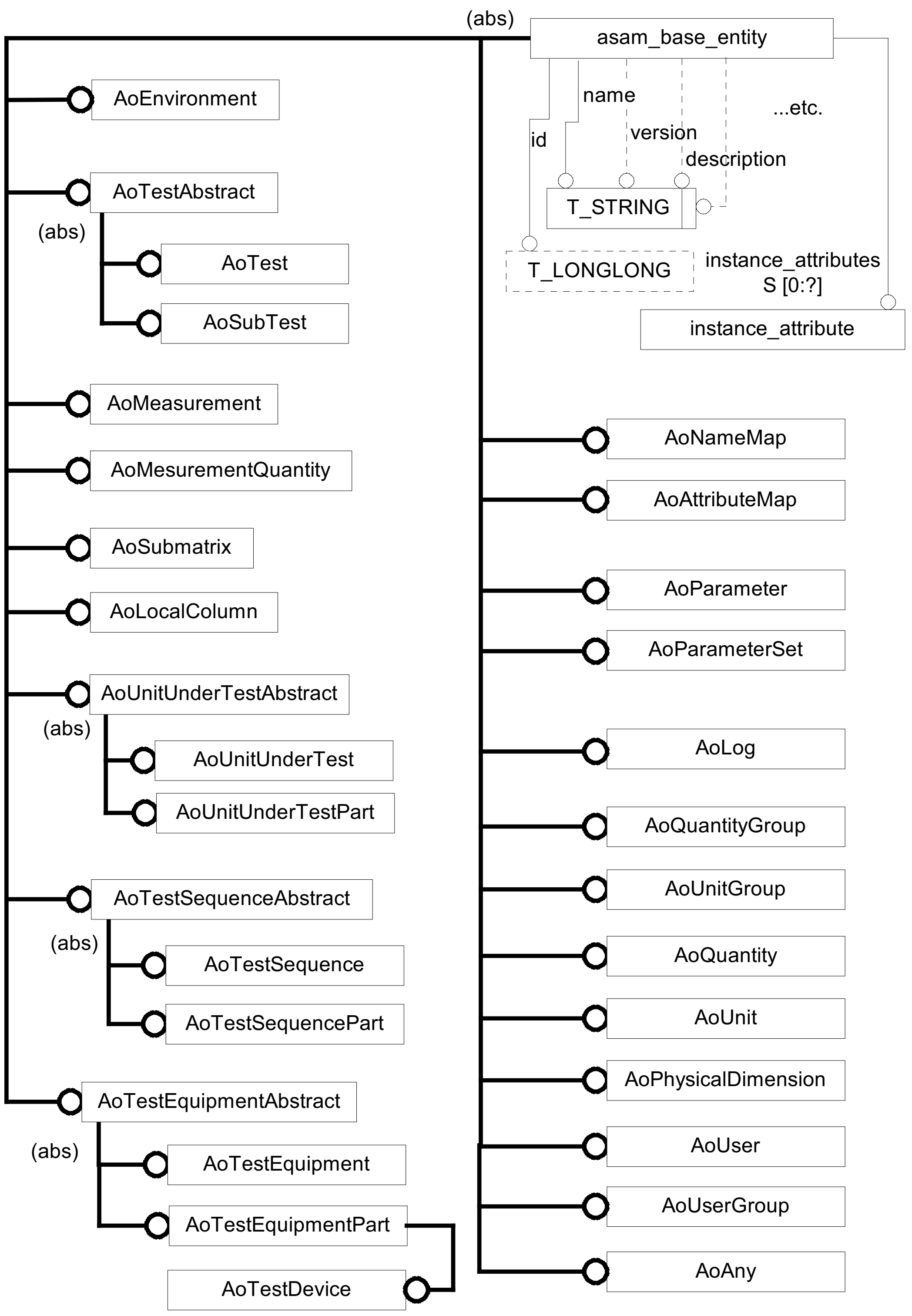
Environment



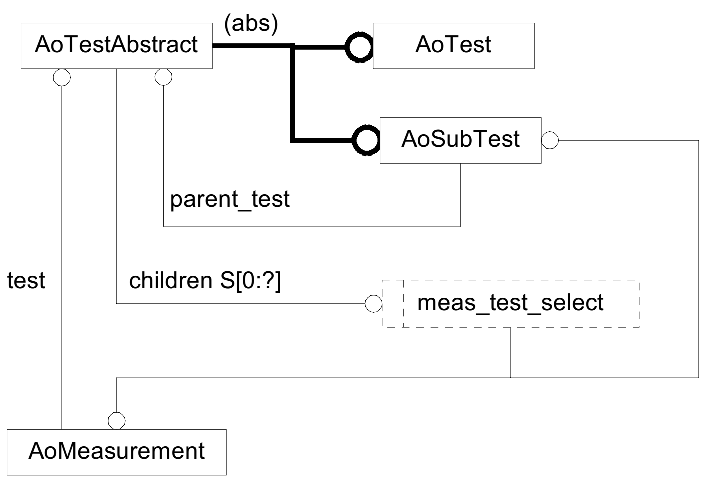
Top Entity and Attributes Objects

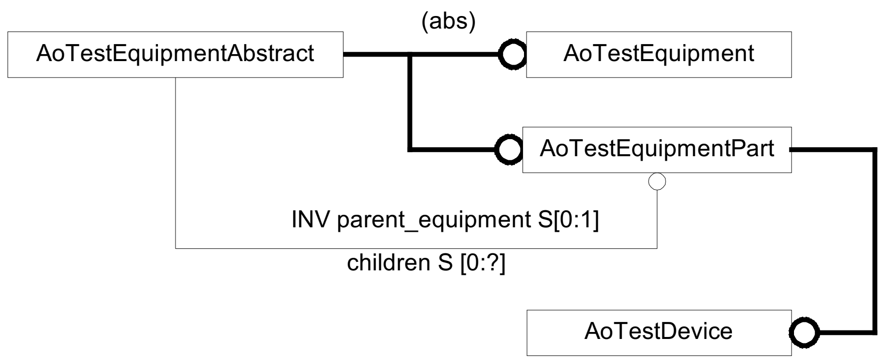


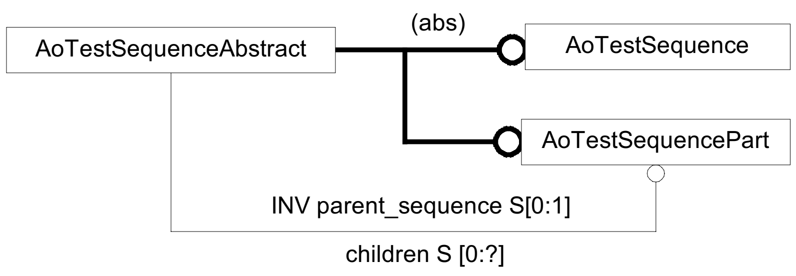
Base Entities

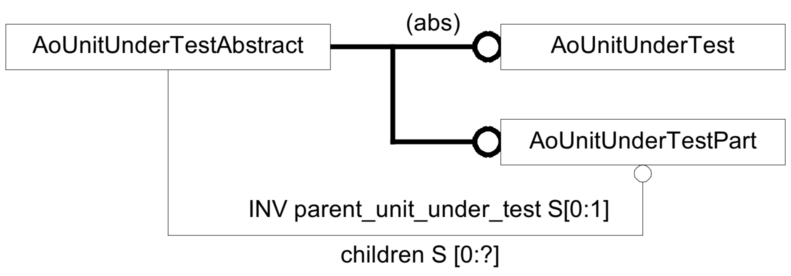


Hierarchical Entities

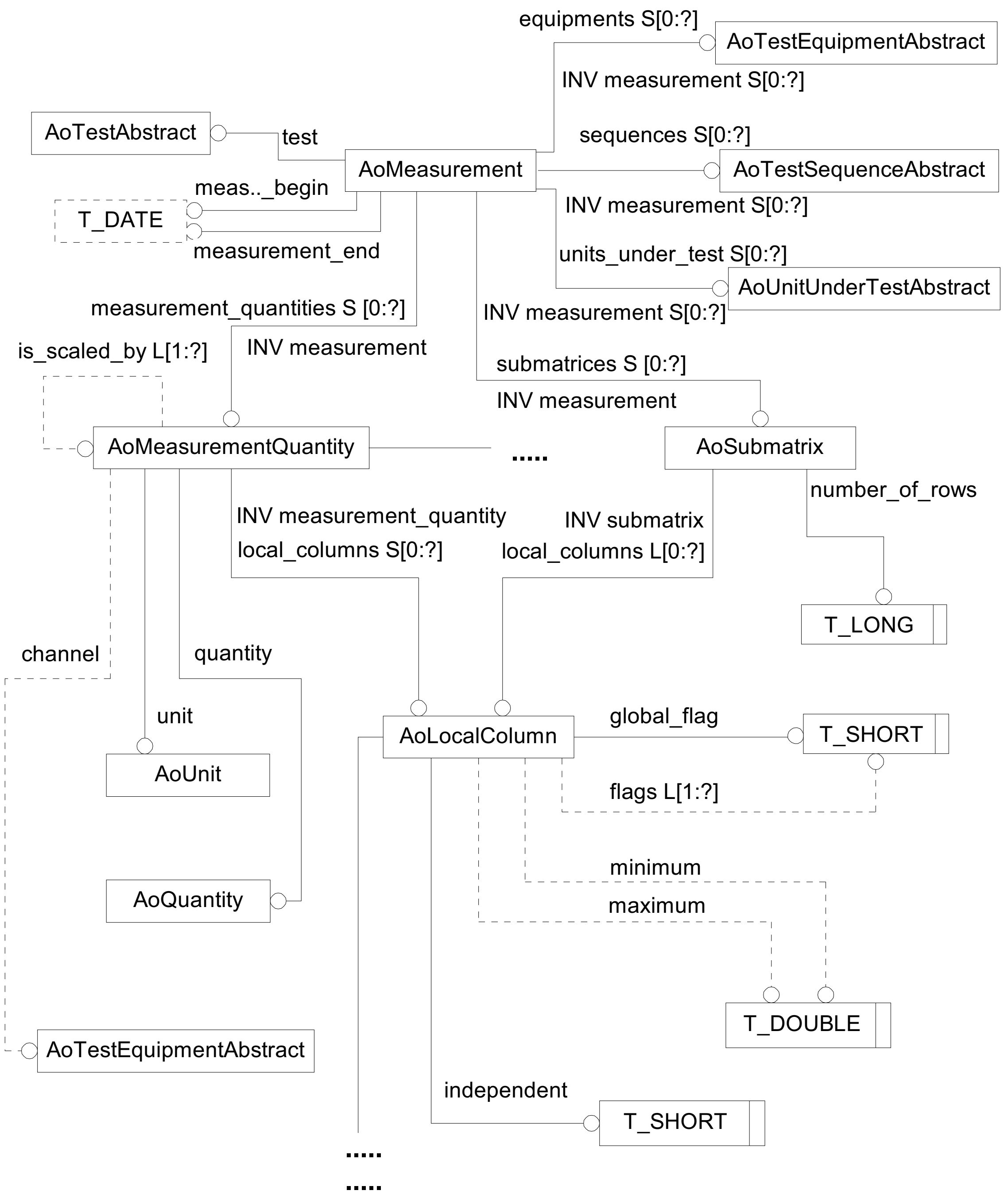




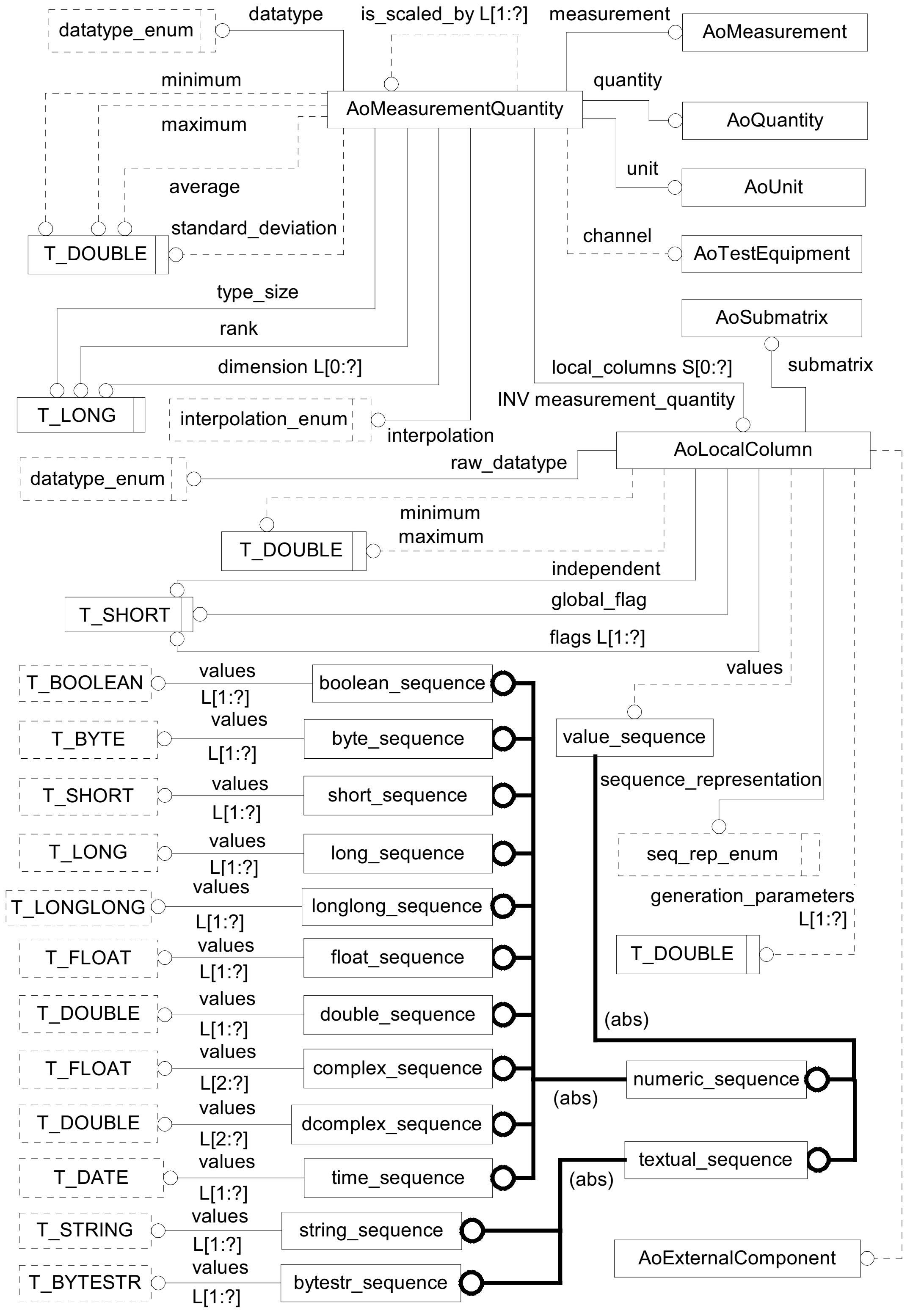




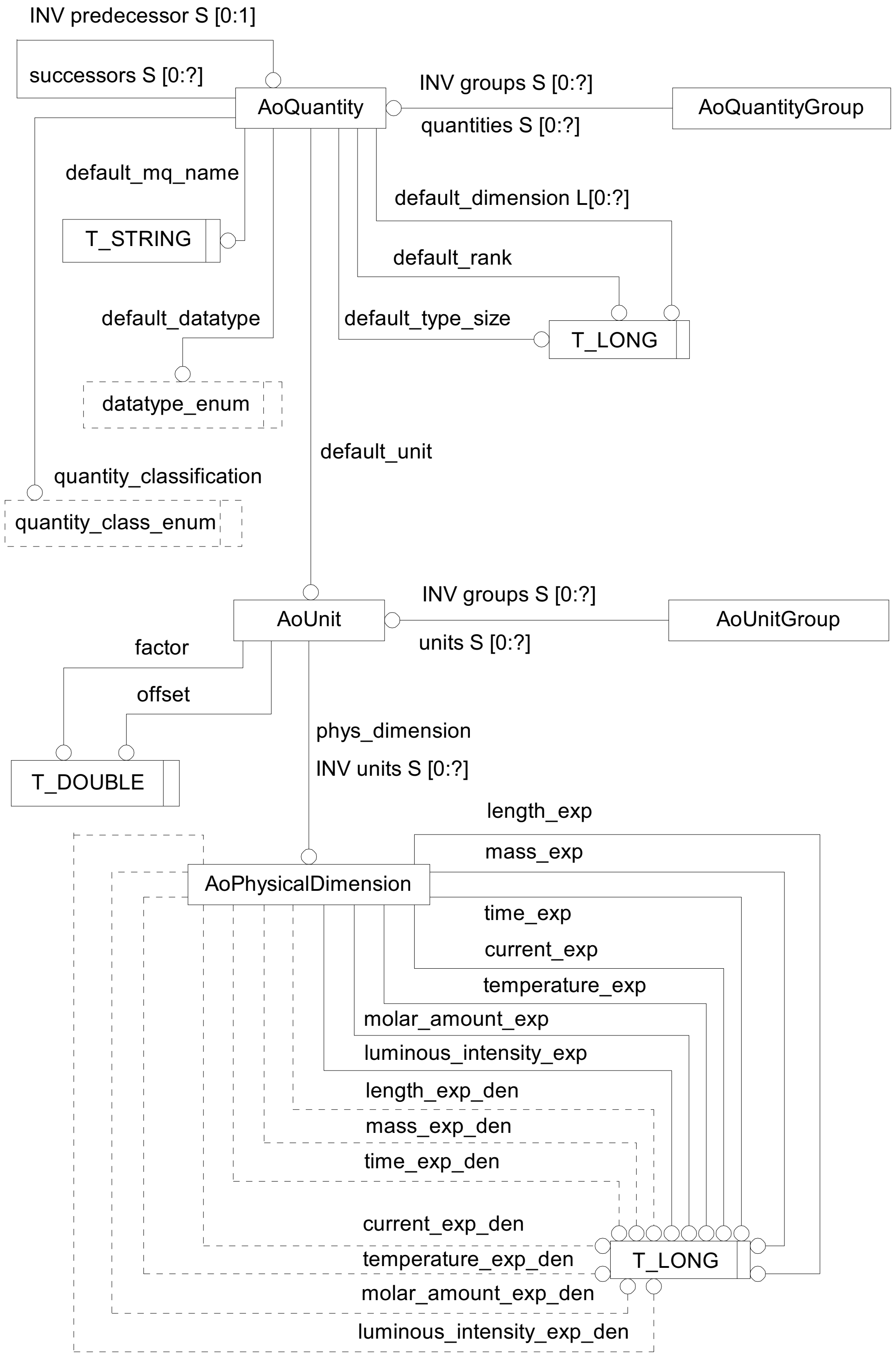
Measurement and Submatrix



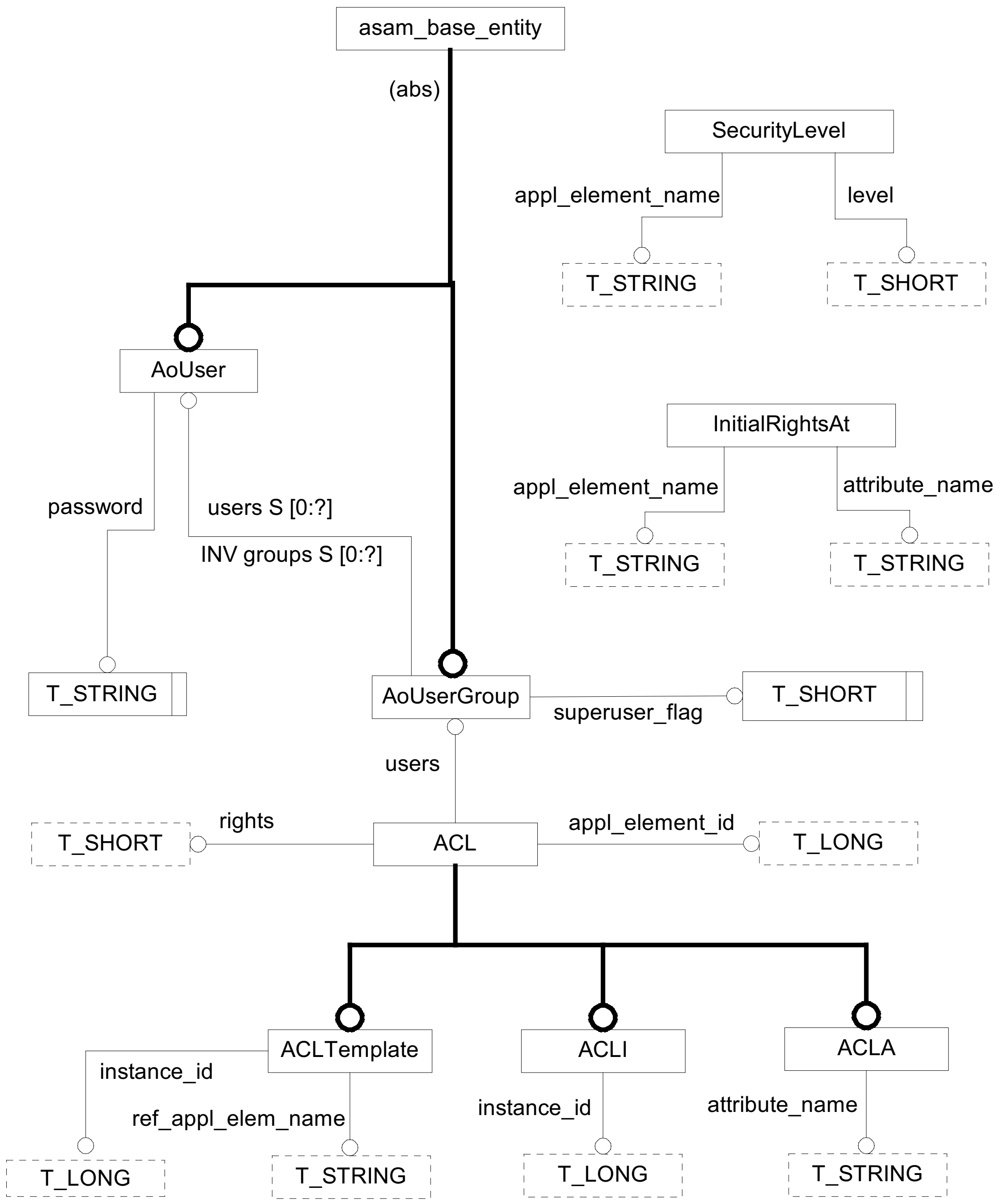
Measurement Quantity and Local Columns



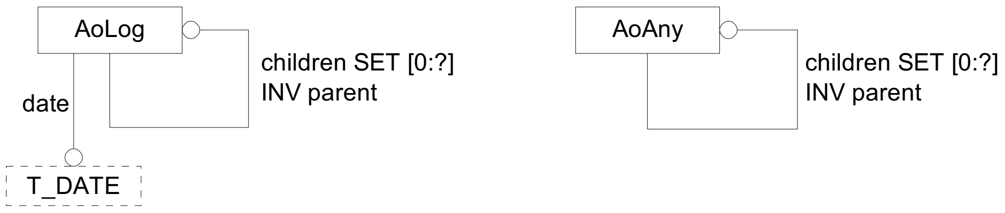
Quantity and Unit

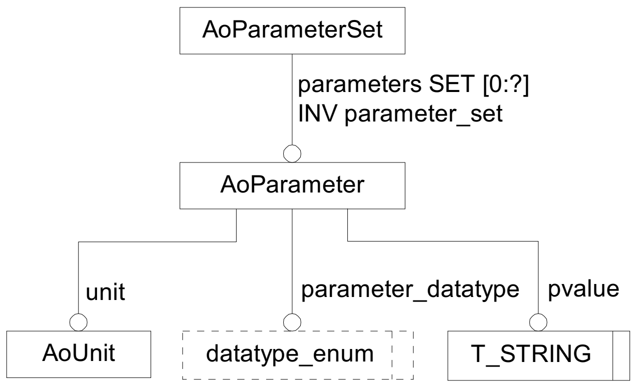


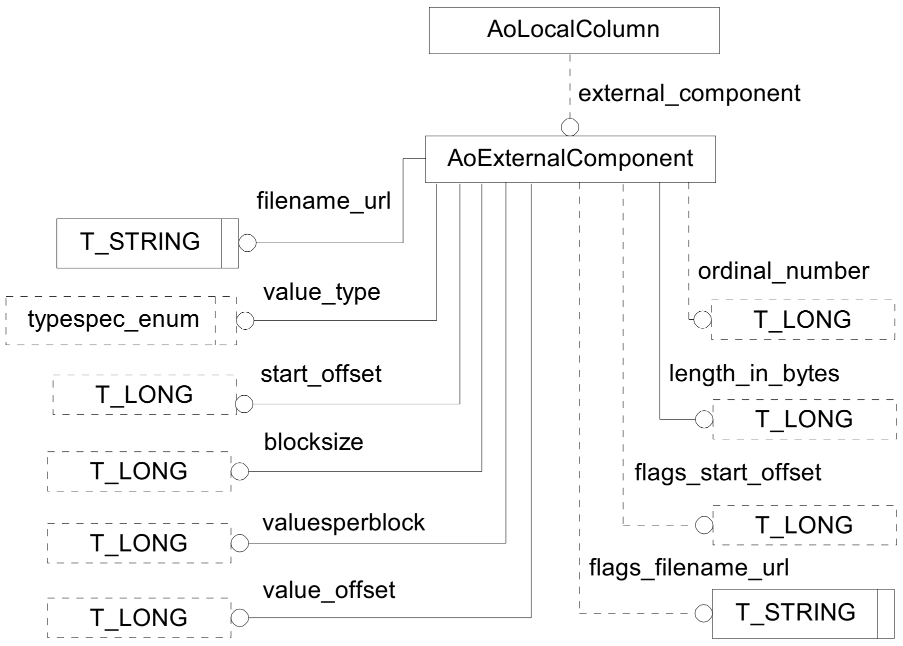
Security


asam_base_entity
(abs)
SecurityLevel
appl_element_name
level
T_STRING
T_SHORT
AoUser
InitialRightsAt
appl_element_name
attribute_name
T_STRING
T_STRING
password
users S [0:?]
INV groups S [0:?]
T_STRING
AoUserGroup
superuser_flag
T_SHORT
users
rights
T_SHORT
ACL
appl_element_id
T_LONG
ACLTemplate
ACLI
ACLA
instance_id
ref_appl_elem_name
instance_id
attribute_name
T_LONG
T_STRING
T_LONG
T_STRING

Parameter, Log, External Component







4.10 REVISION HISTORY

Date Editor	Changes
2003-06 P. Voltmann	New base elements AoExternalComponent, AoNameMap, AoAttributeMap, AoParameterSet, AoParameter and AoLog added New enumerations added Relations of AoNameMap, AoAttributeMap, AoParameterSet and AoParameter extended
2003-10-11 R. Bartz	Several errors have been fixed and explanations have been added to make things clearer
2003-10-17	The FTR meeting agreed to the current text
2003-11-25 R. Bartz	New STEP EXPRESS code and diagrams (that were created by H. Helpenstein) have been included
2003-12-30 R. Bartz	The **Release** version has been created
2004-03 R. Bartz	The attribute 'raw_datatype' has been formally added to AoLocalColumn.
2004-05 R. Bartz	The detailed textual description of the Base Elements has been included (formerly being part of Chapter 2) The explanation of the quantity hierarchy has been improved Some corrections have been made to the Base Element tables
2004-09 R. Bartz	The Base Elements tables have been expanded by adding complete information on datatypes and relation types Some 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 5
ASAM TRANSPORT FORMAT CLASSIC (ATF/CLA)

Version 1.4.1

Association for **S**tandardization of
Automation and **M**easuring Systems

Dated: 30.09.2004

© ASAM e.V.

Status of Document

Reference:	ASAM ODS Version 5.0 ASAM Transport Format Classic
Date:	30.09.2004
Author:	Hans-Joachim Bothe, HighQSoft; Dr. Helmut Helpenstein, National Instruments
Type:	Specification
Doc-ID:	ASAM_ODS_50_CH05_ATF_CLA.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 (odsftr@asam.net) to make sure this issue will be addressed within the next review cycle.

Contents

5 THE ASAM TRANSPORT FORMAT CLASSIC (ATF/CLA) 5-5
5.1 INTRODUCTION 5-5
5.2 THE ATF/CLA FILE 5-7
5.2.1 PURPOSE OF AN ATF/CLA FILE 5-7
5.2.2 STRUCTURE OF AN ATF/CLA FILE 5-8
5.3 DESCRIPTION OF THE ATF/CLA SYNTAX 5-9
5.3.1 OVERVIEW 5-9
5.3.2 DESCRIPTION OF THE CHARACTER SETS USED IN THIS DOCUMENTATION 5-10
5.4 DESCRIPTION OF THE NOTATION USED IN THIS DOCUMENTATION 5-11
5.5 RESERVED ATF/CLA KEYWORDS 5-12
5.6 BRIEF DESCRIPTION OF THE RESERVED ATF/CLA KEYWORDS 5-13
5.6.1 STRUCTURE OF RESERVED ATF/CLA KEYWORDS 5-15
5.7 ATF/CLA SYNTAX DIAGRAMS 5-16
5.7.1 STRUCTURE OF SPACES *(WHITESPACE)* 5-16
5.7.2 STRUCTURE OF SEPARATORS (*SEPARATOR*) 5-17
5.7.3 STRUCTURE OF COMMENTS (*COMMENT*) 5-17
5.7.4 STRUCTURE OF ESCAPE SEQUENCES (*ESCAPE*) 5-18
5.7.5 STRUCTURE OF INTEGERS (*INTEGER*) 5-21
5.7.6 STRUCTURE OF FLOATING-POINT NUMBERS (*REAL*) 5-22
5.7.7 STRUCTURE OF COMPLEX NUMBERS (*COMPLEX*) 5-27
5.7.8 STRUCTURE OF CHARACTER CONSTANTS (*CHARACTER*) 5-27
5.7.9 STRUCTURE OF STRINGS (*STRING*) 5-28
5.7.10 STRUCTURE OF BYTESTREAM (*BYTESTREAM*) 5-29
5.7.11 STRUCTURE OF IDENTIFIERS (*IDENTIFIER*) 5-30
5.7.12 STRUCTURE OF FILENAMES (*FILENAME*) 5-31
5.7.13 PREDEFINED DATA TYPES (*DATATYPE*) 5-33
5.7.14 STRUCTURE OF ENUMERATION 5-35
5.7.15 STRUCTURE OF THE ATF/CLA VERSION IDENTIFIER (*VERSION*) 5-36
5.7.16 STRUCTURE OF THE FILES BLOCK (*FILES*) 5-37
5.7.17 STRUCTURE OF THE INSTRUCTION INCLUDE (*INCLUDE*) 5-38
5.7.18 REFERENCES TO UNITS (*PHYS_UNIT*) 5-39
5.7.19 VALUES OF DATA ATTRIBUTES (*DATAVALUE*) 5-40
5.7.20 STRUCTURE OF APPLICATION ELEMENTS (*APPLELEM*) 5-40
5.7.21 STRUCTURE OF INSTANCE ELEMENTS (*INSTELEM*) 5-42
5.7.22 STRUCTURE OF COMPONENTS (*COMPONENT*) 5-46
5.7.23 THE STRUCTURE OF ENDFILE (*ENDFILE*) 5-55
5.8 OVERVIEW OVER THE STRUCTURE OF AN ATF/CLA FILE (*ATF_OVERVIEW*) 5-56
5.9 SECURITY INFORMATION ON ATF/CLA FILES 5-57
5.9.1 THE USE OF SECURITY INFORMATION ON ATF/CLA 5-57

5.9.2 WRITING SECURITY INFORMATION ON ATF/CLA.......................................5-57
5.10 EXAMPLE FOR AN APPLICATION.......................................5-59
5.11 BIBLIOGRAPHY.......................................5-86
5.12 REVISION HISTORY.......................................5-87
5.12.1 PREVIOUS CHANGES.......................................5-87
5.12.2 KNOWN ISSUES.......................................5-88

Scope

This document describes the classic version of the ASAM Transport Format (ATF/CLA) of ASAM ODS Version 5.0.

Intended Audience

This document is intended for implementers of ASAM ODS Version 5.0. It shall be used as a technical reference with examples how to write ATF/CLA files with the required information used in 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

5 THE ASAM TRANSPORT FORMAT CLASSIC (ATF/CLA)

5.1 INTRODUCTION

ASAM ODS offers the possibility to store measured data in a standardized format. It has a data model with a general structure frame for the modelling of data. It also defines a set of functions for reading and writing the parameter and resulting data, which are to be stored in the data structure.

These functions are based on different kinds of physically storing information. One possibility is storing the data in a commercial data base, while representing the individual application model in data base tables. On the other hand, a specification exists for a file storing system which makes it possible to run ASAM compatible applications without using a data base.

The third possibility to access data with ASAM functions is to represent existing data from a data base or a file system with the help of meta information on an ASAM data structure. By the means of this meta information an existing data storage system becomes compatible to ASAM and works nearly identical to the possibilities mentioned before.

Referring to the information above it is not the only question how to store and backup data in a special data structure, the so called environment, it is also important how to exchange data between different computers, applications and environments.

With identical environments, it is theoretically possible to use functions for the import and export of data from data bases or to exchange files within the respective file systems. As this exchange of data is not supported by the ASAM application there is a risk of affecting the integrity and consistency of the exchanged data.

To avoid such problems in data exchange, ASAM ODS offers a special file format, the ASAM Transport Format (ATF). ATF makes it possible to exchange whole environments or even parts of those between different computers and environments.

ATF is able to transport the required structure and instance information as well as the respective data and to store all this information in a standardized format. The transport format does not guarantee the ability of the reading application to class the information with its own data structures. This can only be guaranteed if the data structures of the reading application are able to pick up the ATF information. On the other hand, the data integrity and compatibility to ASAM is always ensured.

Version 5.0 of the ASAM ODS standard includes two such file formats: the classic ATF (ATF/CLA) and the XML based ATF (ATF/XML). The Introduction (chapter 1) states background and purpose of two such definitions. This chapter describes and explains the classic ASAM Transport Format (ATF/CLA). Chapter 6 is dedicated to the XML based ATF.

5.2 THE ATF/CLA FILE

5.2.1 PURPOSE OF AN ATF/CLA FILE

A file format for transporting or archiving of data must meet other requirements than a file format used for access during the execution of a program (working format).

With working formats as the ASAM file format [PROT] the access performance and the ability of editing is paramount. As it is normally not really possible to insert data in a file but to append them, the data in the ASAM file storage are structured that way, that additional data are written at the end of the individual file.

A data transport format does not have these requirements, because the respective file is read and written completely in one step and there is no need for time critical access. Transport formats must meet other requirements, for example the portability between different computer systems.

To be accepted by the user and for effective maintenance the structure of the transport file must be simple and easy to survey. Therefore, an easy to interpret syntax facilitates the reading and writing of ATF/CLA files even for programs not based on ASAM.

Independent from the above mentioned facts ATF/CLA must be able to store all information defined in the ASAM data model in an unique way. This refers not only to the structure information but also to the data itself.

The first version of the ATF/CLA file will be a pure ASCII format to enable a data transfer on different computer platforms. In this case the ASCII standard is to be considered which only defines the first 127 characters. If mutated vowels (umlauts) are used, the possibility of incompatibilities is given, because those characters are interpreted differently on different systems.

Therefore, special characters should not be used even in plain text. In this documentation you will find a proposal for the solution of transferring mutated vowels.

With respect to the fast growing worldwide nets it is planned for a later version of ATF/CLA to use the „Unicode character encoding“. The Unicode character set is a worldwide agreed upon 16 bit character set, which implements a superset of the ASCII character set. For more information refer to the bibliography under [UNICODE].

The ATF/CLA file is to be seen as a logical file. This means, that an ATF/CLA file may be distributed physically over more than one file. This is especially interesting when storing and transporting the structure information separately from the instance information.

Very often it is likely not to transport the measured and mass data in ASCII format, because the amount of data may become very large. In this case it is possible to store the mass data in a binary format, whereas the different computer architectures are to be respected (little/big endian, float formats).

The main file keeping the references of the subfiles should get the extension „.atf“ to be recognised as ATF/CLA file. All the other files may have individual names and extensions.

5.2.2 STRUCTURE OF AN ATF/CLA FILE

The ATF/CLA file is built up in three different block types, they are described in section 5.7:

- The Files section
- The Application Elements section
- The Instance Elements section

The file starts with the keyword ATF_FILE followed by a version number which shows the actual version of the ATF/CLA specification the file is based on. The three main building blocks are FILES, APPLELEM and INSTELEM. The end of ATF/CLA definitions in a file is marked by the keyword ATF_END. For error tracking purposes, the node name, username and program name of the generating application should be added as a comment at the beginning of the ATF/CLA file.

The intention of the ATF/CLA file is to show the data as they exist in the data source, any interpretation shall be performed at the destination. Therefore ATF/CLA-Files should only contain metadata and data which are available in the data source. Elements and attributes that are not available in a data source shall not be written to the ATF/CLA file even if they belong to the ASAM ODS base model.

Consequently all application elements that are derived from base elements and are instantiated in the file must be described in the Application Elements section. Except for a few security entities (see section 5.9, “Security Information on ATF/CLA files”) the instantiation of entities from the base model is not allowed in ASAM ODS. Therefore apart from these all other entities are application elements.

5.3 DESCRIPTION OF THE ATF/CLA SYNTAX

5.3.1 OVERVIEW

This description of the syntax is intended to define the ASAM transport format in an unique way. The object of this description is to reduce ambiguities and vaguenesses to a minimum or even to avoid those when generating ATF/CLA files.

The syntax of an ATF/CLA file follows partly the syntax of C and C++. To enhance the readability of the file, the use of braces was reduced to a minimum. To implement the function of braces, every block keyword has a respective end-of-block keyword.

Like in C all „whitespaces“ (spaces, tabs and form feeds) outside of strings are ignored or interpreted as separators. This requires all logical lines within an ATF/CLA file to be terminated by a semicolon (;). In this way it is possible, to separate very long logical lines into different physical lines.

The names of identifiers have to follow the ASAM standards. Uppercase and lowercase letters are distinguished.

For the description of regular expressions and constructs used in ATF/CLA files syntax diagrams are used. These syntax diagrams are easy to understand without the knowledge of machine and graph theories.

A syntax diagram is always read from left to right. At branch points the leftmost path is checked first. Paths generating loops (showing backward chainings in the syntax diagram) are marked with an arrow at their end points. These paths must not be followed against the direction of the arrow.

Syntax diagrams may be nested as required. Names of syntax diagrams are written in *italic*. Greek letters in uppercase refer to character sets, greek letters in lowercase refer to the allowed amount of characters of the respective character set.

Keywords are always in uppercase letters. *Identifiers, strings* and *characters* are not converted to uppercase. Uppercase and lowercase letters may be used individually.

5.3.2 DESCRIPTION OF THE CHARACTER SETS USED IN THIS DOCUMENTATION

A	=		Complete printable character set (32 to 126and 128 to 255)
B	=		Not-allowed characters
X	=		Complete non-printable character set (0 to 31 and 127)
Δ	=	{0-9}	Numbers (48 to 57)
H	=	{0-9, A-F, a-f}	Hexadecimal numbers (48 to 57, 65 to 70, 97 to 102)
Λ	=	{A-Z, a-z}	Letters (65 to 90, 97 to 122)
N	=		Non-alphanumeric characters (32 to 47, 58 to 64, 91 to 96, 123 to 126)

Non-printable characters are shown in this documentation as follows (not to be confused with escape sequences in strings):

```
<NUL>, <SOH>, <STX>, <ETX>, <EOT>, <ENQ>, <ACK>, <BEL>,
<BS> , <HT> , <LF> , <VT> , <FF> , <CR> , <SO> , <SI> ,
<DLE>, <DC1>, <DC2>, <DC3>, <DC4>, <NAK>, <SYN>, <ETB>,
<CAN>, <EM> , <SUB>, <ESC>, <FS> , <GS> , <RS> , <US> ,
<DEL>
```

Due to readability, tabs and spaces are shown as follows: `<HT>, <SP>`

5.4 DESCRIPTION OF THE NOTATION USED IN THIS DOCUMENTATION

- Complete printable character set:

  α = Exactly one character of the character set A.

  α^n = Exactly n characters of the character set A.

  α^+ = Arbitrary amount of characters, at least one character.

  α^* = Arbitrary amount of characters. „No character“ is also applicable ($\alpha^+|\varepsilon$).

- Complete not-allowed character set:

  β = Exactly one character of the character set B.

  β^n = Exactly n characters of the character set B.

  β^+ = Arbitrary amount of characters, at least one character.

  β^* = Arbitrary amount of characters. „No character“ is also applicable ($\beta^+|\varepsilon$).

- Complete non-printable character set:

  χ = Exactly one character of the non-printable character set X.

  χ^n = Exactly n characters of the nonprintable character set X.

  χ^+ = Arbitrary amount of characters, at least one character.

  χ^* = Arbitrary amount of characters. „No character“ is also applicable ($\chi^+|\varepsilon$).

- Numbers:

  δ = Exactly one number of the character set Δ.

  δ^n = Exactly n numbers of the character set Δ.

  δ^+ = Arbitrary amount of numbers, at least one number.

  δ^* = Arbitrary amount of numbers. „No number“ is also applicable ($\delta^+|\varepsilon$).

- Hexadecimal Numbers:

  η = Exactly one number of the character set H.

  η^n = Exactly n numbers of the character set H.

  η^+ = Arbitrary amount of numbers, at least one number.

  η^* = Arbitrary amount of numbers. „No number“ is also applicable ($\eta^+|\varepsilon$).

- Letters:

  λ = Exactly one letter of the character set Λ.

  λ^n = Exactly n letters of the character set Λ.

  λ^+ = Arbitrary amount of letters, at least one letter.

  λ^* = Arbitrary amount of letters ($\lambda^+|\varepsilon$).

- Non-alphanumeric characters:

ν	=	Exactly one character of the character set N.
ν^n	=	Exactly n characters of the character set N.
ν^+	=	Arbitrary amount of characters, at least one character.
ν^*	=	Arbitrary amount of characters ($\nu^+|\varepsilon$).

In the above descriptions ε stands for the empty set.

5.5 RESERVED ATF/CLA KEYWORDS

The following keywords are reserved in ATF/CLA and must not be used as identifier for application and instance elements.

APPLATTR	DT_CFLOAT	IEEEFLOAT4
APPLELEM	DT_DATE	IEEEFLOAT8
ATF_END	DT_DOUBLE	INCLUDE
ATF_FILE	DT_FLOAT	INFINITY
BASEATTR	DT_LONG	INIOFFSET
BASETYPE	DT_LONGLONG	INSTELEM
BLOCKSIZE	DT_SHORT	MANY
CARDINALITY	DT_STRING	NAN
COMPONENT	DT_UNKNOWN	REF_TO
DATATYPE	DESCRIPTION	REF_TYPE
DT_BLOB	ENDAPPLELEM	TRUE
DT_BOOLEAN	ENDFILES	UNDEFINED
DT_BYTE	ENDINSTELEM	VALOFFSETS
DT_BYTESTR	FALSE	VALPERBLOCK
DT_CDOUBLE	FILES	

5.6 BRIEF DESCRIPTION OF THE RESERVED ATF/CLA KEYWORDS

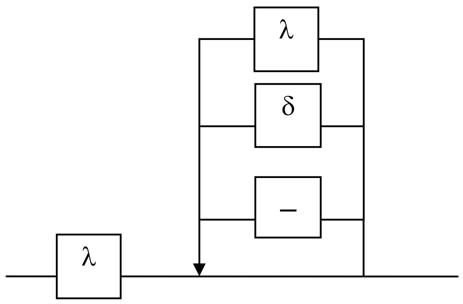
APPLATTR	Definition of an application attribute.
APPLELEM	Starts the definition of an application element.
ATF_END	Indicates the end of an ATF/CLA file. Further information will not be checked by the parser.
ATF_FILE	Starts an ATF/CLA file and must exist at the beginning of the main file.
BASEATTR	Specifies the base type of an application attribute.
BASETYPE	Specifies the base type of an application element.
BLOCKSIZE	Size of a block on a binary dataset
CARDINALITY	Specifies, how many values may be given for an attribute
COMPONENT	Reference on an external file
DATATYPE	Specifies the data type of application and instance attributes.
DT_BLOB	Data type for binary large object blocks
DT_BOOLEAN	Data type Boolean, allowed values are TRUE and FALSE.
DT_BYTE	Data type for the representation of character constants.
DT_BYTESTR	Data type for byte streams
DT_COMPLEX	Data type for the representation of complex single precision floating-point numbers (32 bit for real and imaginary part each).
DT_DATE	Data type for date/time
DT_DCOMPLEX	Data type for the representation of complex double precision floating-point numbers (64 bit for real and imaginary part each).
DT_DOUBLE	Data type for the representation of double precision floating-point numbers (64 bit).
DT_LONG	Data type for the representation of integers with 32 bit.
DT_LONGLONG	Data type for the representation of integers with 64 bit.
DT_FLOAT	Data type for the representation of single precision floating-point numbers (32 bit).
DT_SHORT	Data type for the representation of integers with 16 bit.

DT_STRING	Data type for the representation of strings (text constants).
ENDAPPLELEM	Ends the definition of an application element.
ENDFILES	Ends the list of physical files belonging to a logical ATF/CLA file.
ENDINSTELEM	Ends the definition of an instance element.
FALSE	Value „False“ for data type Bool.
FILES	The block Files is used to list all physical files belonging to a logical file.
IEEEFLOAT4	Binary representation of a 4 byte float number
IEEEFLOAT8	Binary representation of a 8 byte float number
INCLUDE	Inserts a specified file instead an Include instruction.
INFINITY	Marks the value "Infinity".
INIOFFSET	Initial offset in bytes from the begin of a binary file to the first block.
INSTELEM	Starts the definition of an instance element.
MANY	An arbitrary number of elements.
NAN	Not a Number.
REF_TO	Reference definition in application element definition.
REF_TYPE	Specification of an application element in an instantiated attribute.
TRUE	Value „TRUE“ for the data type Boolean.
UNDEFINED	Keyword for resetting of values to the status „not defined“.
VALOFFSETS	Offsets in byte from the beginning of a block to the values on a binary file.
VALPERBLOCK	Number of values per block in binary file.

5.6.1 STRUCTURE OF RESERVED ATF/CLA KEYWORDS

The keywords in ATF/CLA files always have to start with a letter. The letter may be followed by an arbitrary amount of letters, numbers and underscores. All other characters not mentioned here are not allowed in keywords.

Syntax diagram ***keyword***:



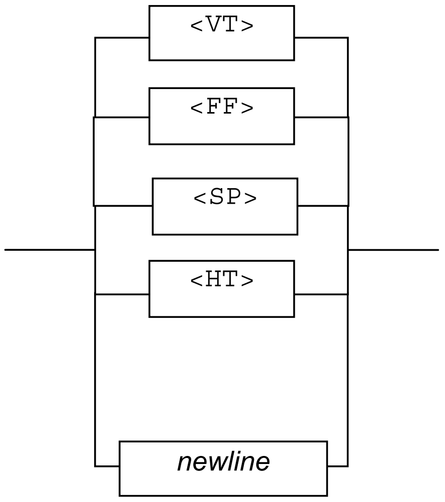
This rule does not have to be implemented. It is just a rule for the definition of ATF/CLA keywords, because keywords are taken literally from the list above.

5.7 ATF/CLA SYNTAX DIAGRAMS

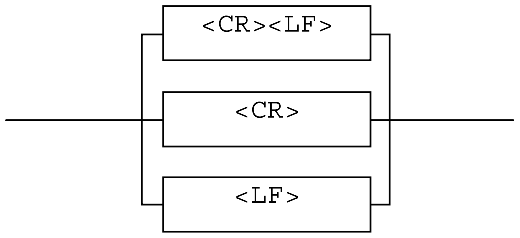
5.7.1 STRUCTURE OF SPACES (*WHITESPACE*)

A carriage return is defined as `<CR><LF>`, `<CR>` and `<LF>`. Please note: The (normally not used) representation `<LF><CR>` is counted as two carriage returns. This effects line numbering in error reports only.

Syntax diagram ***Whitespace***



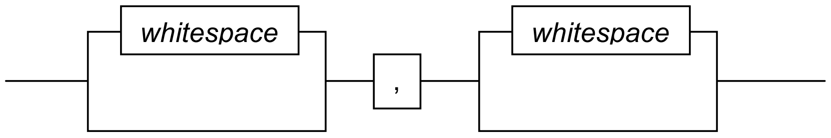
Syntax diagram Newline



The elements in the syntax diagram *whitespace* will be ignored automatically at any position except within strings ("...") or character constants (‘...’). Please find further information in chapter 5.3.1, Overview. Whitespaces are shown as follows:

○	A whitespace may be included.
●	At least one whitespace must be included.

5.7.2 STRUCTURE OF SEPARATORS (*SEPARATOR*)



Separators are shown as follows:

☐ A separator <u>may be</u> included.

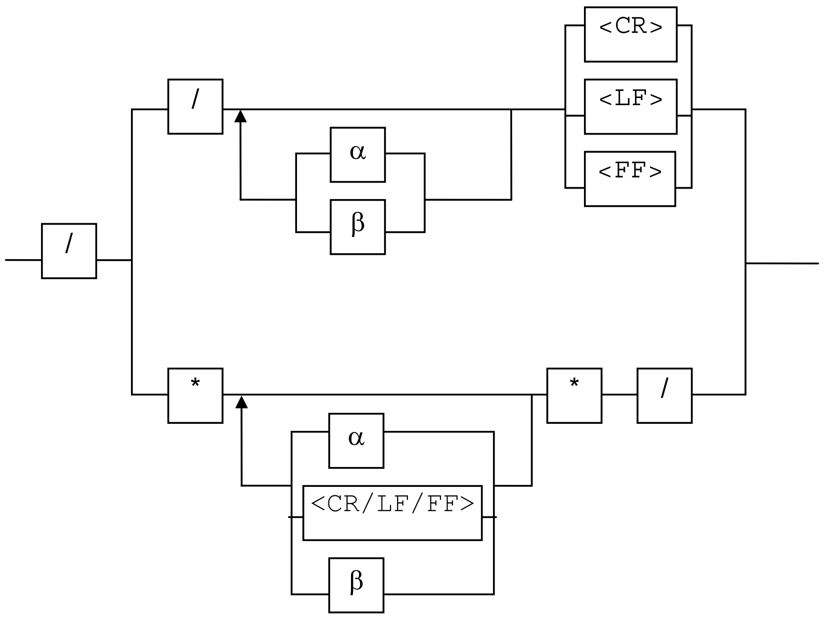
■ At least one separator <u>must be</u> included.

5.7.3 STRUCTURE OF COMMENTS (*COMMENT*)

Comments are enclosed in „/*“ and „*/“ or started with „//“ as comment to the end of the (physical) line. The following rules apply:

- Comments must not be nested.
- Comment markings within strings or character constants are not interpreted respectively.
- „/*“ and „*/“ have no meaning after „//“.
- „//“ has no meaning in comments enclosed by „/*“ and „*/“.
- Comments are classified as separator meaning e.g. the sequence "text /* ... */ constant" will be interpreted as two identifiers separated by a separator.

<u>Syntax diagram ***comment***:</u>



5.7.4 STRUCTURE OF ESCAPE SEQUENCES (*ESCAPE*)

Escape sequences are allowed within strings, character constants and identifiers. Those sequences will be converted to one byte during the reading process. The Escape sequences have the following meaning:

- Insertion of non-printable characters (e.g. new line, form feed etc.) in strings and character constants.
- Encapsulation of special characters (e.g. colon, semicolon, parenthesis etc.) in identifiers, when these special characters could lead to ambiguities in the syntax of an ATF/CLA file.
- Unique transfer of characters outside the 7 bit ASCII character set, which would be interpreted in a different way on different machines (e.g. German umlauts).

Because Escape sequences affect the readability of ATF/CLA files, their use should be reduced as much as possible.

During the interpretation of Escape sequences no further substitution is done, if the Escape sequence is already the result of an interpretation of another Escape sequence. This makes it easier for the construction of an interpreter, because the read pointer does not have to be positioned backwards.

Escape sequences are generally started by „\“. If this character is to be used in strings or character constants it must be doubled: „\\“.

The following Escape sequences for printable characters are allowed in strings, character constants and identifiers in ATF/CLA files:

\<sp>	Space (needed in identifiers)
\ν	All non-alphanumeric characters, e.g. “ inside a string
\0xdd	Hexadecimal representation of an arbitrary printable character

No longer needed, because extended character set is allowed in strings (included to understand older documents and files):

\<ae>	Umlaut ä
\<Ae>	Umlaut Ä
\<oe>	Umlaut ö
\<Oe>	Umlaut Ö
\<ue>	Umlaut ü
\<Ue>	Umlaut Ü
\<sz>	Special character ß
\<^0>	Degree (e.g. °Celsius)

Syntax diagram ***escape***:

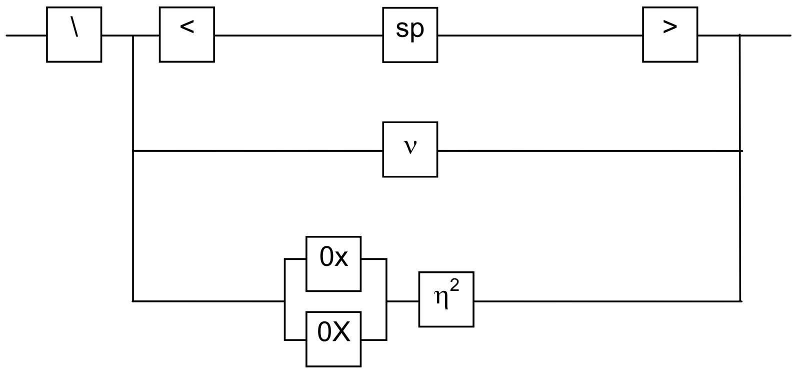


Table of umlaut codings given here for information only

	ISO
ä	228
Ä	196
ö	246
Ö	214
ü	252
Ü	220
ß	223
º	176

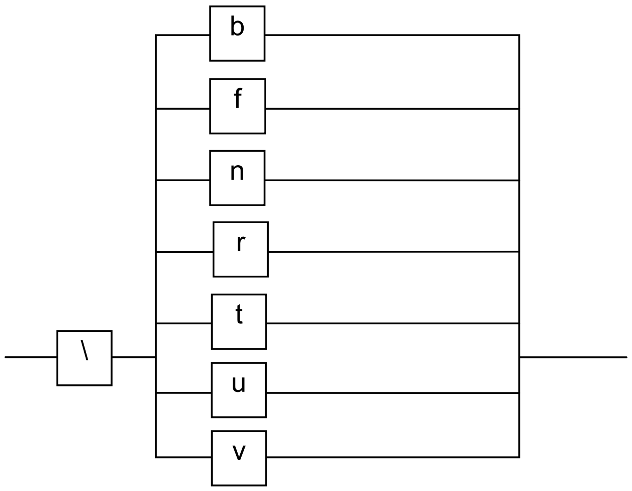
The following additional Escape sequences for non-printable characters are allowed in strings, character constants and identifiers in ATF/CLA files:

Sequence	Code	Description	
\b	8	Backspace	<BS>
\f	12	Form feed	<FF>
\n	10	New line	<LF>
\r	13	Carriage return	<CR>
\t	9	Tab	<HT>
\u		Reserved for Unicode extensions	
\v	11	Vertical Tab	<VT>

All other substitute representations must be done by using the normal escape sequences.

For identifiers beginning with a number (0 ... 9) the substitute representation \0 to \9 must be used. Therefore, the value <NUL> can only be shown with the hexadecimal representation \0x00.

Syntax diagram ***escapenonprint***:



Because future versions of ATF/CLA may work with 16 bit characters, care should be taken during implementation. While reading an ATF/CLA file it should be checked, whether the ATF/CLA version number is not higher than the version used during the implementation of the reading program.

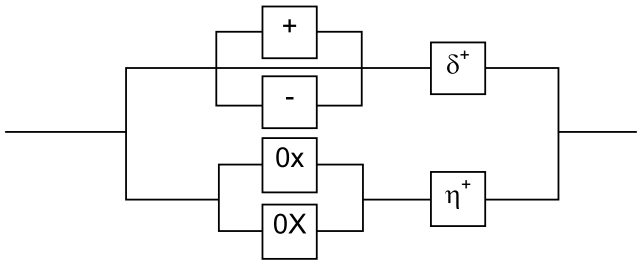
5.7.5 STRUCTURE OF INTEGERS (*INTEGER*)

Integers can be represented in ATF/CLA with a length of 8 (*byte*), 16 (*int2*), 32 (*int4*) and 64 (*int8*) bit (1, 2, 4 and 8 Byte). When using a length of 64 bit it has to be considered that a lot of today's computers only work with 32 bit. On machines of this type it is not possible without proper emulation to process 64 bit values which would lead to an error message.

The data type *byte* is generally interpreted as unsigned. During the recognition of byte data the parser should be able to accept negative numbers (-1 is equal to 255 etc.).

There will be no octal representation in ATF/CLA, because octal values are easily convertable into hexadecimal values. Leading zeros are ignored.

Syntax diagram ***integer***:



5.7.6 STRUCTURE OF FLOATING-POINT NUMBERS (*REAL*)

Single precision floating-point numbers (*real4*) are represented in ATF/CLA with 32 bit, double precision (*real8*) with 64 bit. The exact range of numbers depends on the architecture of the machine and the division of the machine words into mantissa and exponent.

The preferred ATF/CLA format is the „IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std. 754-1985 (IEEE; New York)“. This section describes how floating-point data is stored in memory. It summarizes the precisions and ranges of the different IEEE storage formats.

A floating-point format is a data structure specifying the fields that comprise a floating-point numeral, the layout of those fields, and their arithmetic interpretation.

A floating-point storage format specifies how a floating-point format is stored in memory. The IEEE standard defines the formats, but it leaves to implementors the choice of storage formats.

Assembly language software sometimes relies on using the storage formats, but higher level languages usually deal only with the linguistic notions of floating-point data types. These types have different names in different high-level languages, and correspond to the IEEE formats. IEEE Standard 754 specifies exactly the single and double floating-point formats, and it defines a class of extended formats for each of these two basic formats.

The following sections describe in detail each of the three storage formats used for the IEEE floating-point formats.

Single Format

The IEEE single format consists of three fields: a 23-bit fraction, f; an 8-bit biased exponent, e; and a 1-bit sign, s. These fields are stored contiguously in one 32-bit word. Bits 0:22 contain the 23-bit fraction, f, with bit 0 being the least significant bit of the fraction and bit 22 being the most significant; bits 23:30 contain the 8-bit biased exponent, e, with bit 23 being the least significant bit of the biased exponent and bit 30 being the most significant; and the highest-order bit 31 contains the sign bit, s.

The following table shows the correspondence between the values of the three constituent fields s, e and f, on the one hand, and the value represented by the single- format bit pattern on the other; u means don't care, that is, the value of the indicated field is irrelevant to the determination of the value of the particular bit patterns in single format.

Table: Values Represented by Bit Patterns in IEEE Single Format

Single-Format Bit Pattern	Value
0 < e < 255	(-1)s x 2e-127 x 1.f (normal numbers)
e = 0; f != 0 (at least one bit in f is nonzero)	(-1)s x 2-126 x 0.f (subnormal numbers)
e = 0; f = 0 (all bits in f are zero)	(-1)s x 0.0 (signed zero)
s = 0; e = 255; f = 0 (all bits in f are zero)	+Infinity (positive infinity)
s = 1; e = 255; f = 0 (all bits in f are zero)	-Infinity (negative infinity)
s = u; e = 255; f != 0 (at least one bit in f is nonzero)	NaN (Not-a-Number)

Notice that when e < 255, the value assigned to the single format bit pattern is formed by inserting the binary radix point immediately to the left of the fraction's most significant bit, and inserting an implicit bit immediately to the left of the binary point, thus representing in binary positional notation a mixed number (whole number plus fraction, wherein 0 <= fraction < 1).

The mixed number thus formed is called the single-format significand. The implicit bit is so named because its value is not explicitly given in the single- format bit pattern, but is implied by the value of the biased exponent field.

For the single format, the difference between a normal number and a subnormal number is that the leading bit of the significand (the bit to left of the binary point) of a normal number is 1, whereas the leading bit of the significand of a subnormal number is 0. Single-format subnormal numbers were called single-format denormalized numbers in IEEE Standard 754.

The 23-bit fraction combined with the implicit leading significand bit provides 24 bits of precision in single-format normal numbers.

Examples of important bit patterns in the single-storage format are shown in the next table. The maximum positive normal number is the largest finite number representable in IEEE single format. The minimum positive subnormal number is the smallest positive number representable in IEEE single format. The minimum positive normal number is often referred to as the underflow threshold. (The decimal values for the maximum and minimum normal and subnormal numbers are approximate; they are correct to the number of digits shown.)

Table: Bit Patterns in Single-Storage Format and their IEEE Values

Common Name	Bit Pattern (Hex)	Decimal Value
+ 0	00000000	0.0
- 0	80000000	-0.0
1	3f800000	1.0
2	40000000	2.0
Maximum normal number	7f7fffff	3.40282347e+38
Minimum positive normal number	00800000	1.17549435e-38
Maximum subnormal number	007fffff	1.17549421e-38
Minimum positive subnormal number	00000001	1.40129846e-45
+ ∞	7f800000	Infinity
- ∞	ff800000	-Infinity
Not-a-Number	7fc00000	NaN

A NaN (Not a Number) can be represented with any of the many bit patterns that satisfy the definition of a NaN. The hex value of the NaN shown in the above table is just one of the many bit patterns that can be used to represent a NaN.

Double Format

The IEEE double format consists of three fields: a 52-bit fraction, f; an 11-bit biased exponent, e; and a 1-bit sign, s. These fields are stored contiguously in two successively addressed 32-bit words.

In the SPARC architecture, the higher address 32-bit word contains the least significant 32 bits of the fraction, while in the Intel and PowerPC architectures the lower address 32-bit word contains the least significant 32 bits of the fraction.

If we denote f[31:0] the least significant 32 bits of the fraction, then bit 0 is the least significant bit of the entire fraction and bit 31 is the most significant of the 32 least significant fraction bits.

In the other 32-bit word, bits 0:19 contain the 20 most significant bits of the fraction, f[51:32], with bit 0 being the least significant of these 20 most significant fraction bits, and bit 19 being the most significant bit of the entire fraction; bits 20:30 contain the 11-bit biased exponent, e, with bit 20 being the least significant bit of the biased exponent and bit 30 being the most significant; and the highest-order bit 31 contains the sign bit, s.

The values of the bit patterns in these three fields determine the value represented by the overall bit pattern.

The following table shows the correspondence between the values of the bits in the three constituent fields, on the one hand, and the value represented by the double- format bit pattern on the other; u means don't care, because the value of the indicated field is irrelevant to the determination of value for the particular bit pattern in double format.

Table: Values Represented by Bit Patterns in IEEE Double Format

XXX Double-Format Bit Pattern	Value
0 < e < 2047	(-1)s x 2e-1023 x 1.f (normal numbers)
e = 0; f != 0 (at least one bit in f is nonzero)	(-1)s x 2-1022 x 0.f (subnormal numbers)
e = 0; f = 0 (all bits in f are zero)	(-1)s x 0.0 (signed zero)
s = 0; e = 2047; f = 0 (all bits in f are zero)	+Infinity (positive infinity)
s = 1; e = 2047; f = 0 (all bits in f are zero)	-Infinity (negative infinity)
s = u; e = 2047; f != 0 (at least one bit in f is nonzero)	NaN (Not-a-Number)

Notice that when e < 2047, the value assigned to the double-format bit pattern is formed by inserting the binary radix point immediately to the left of the fraction's most significant bit, and inserting an implicit bit immediately to the left of the binary point. The number thus formed is called the significand. The implicit bit is so named because its value is not explicitly given in the double- format bit pattern, but is implied by the value of the biased exponent field.

For the double format, the difference between a normal number and a subnormal number is that the leading bit of the significand (the bit to the left of the binary point) of a normal number is 1, whereas the leading bit of the significand of a subnormal number is 0. Double-format subnormal numbers were called double-format denormalized numbers in IEEE Standard 754.

The 52-bit fraction combined with the implicit leading significand bit provides 53 bits of precision in double-format normal numbers.

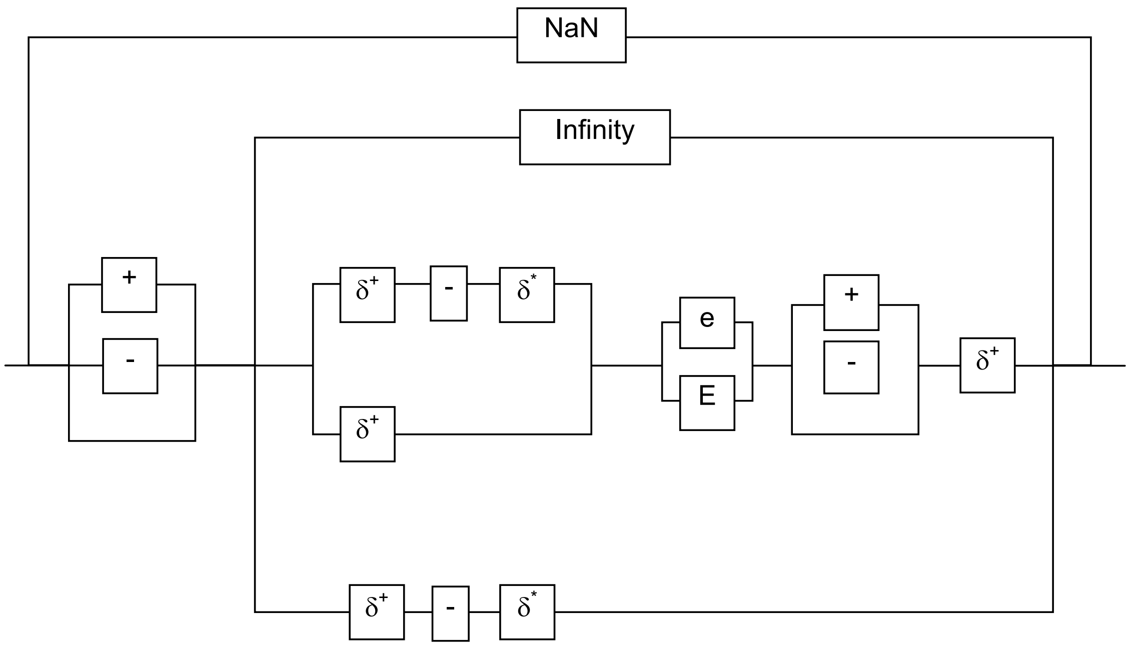
Examples of important bit patterns in the double-storage format are shown in the next table. The bit patterns in the second column appear as two 8-digit hexadecimal numbers. For the SPARC architecture, the left one is the value of the lower addressed 32-bit word, and the right one is the value of the higher addressed 32-bit word, while for the Intel and PowerPC architectures, the left one is the higher addressed word, and the right one is the lower addressed word. The maximum positive normal number is the largest finite number representable in the IEEE double format. The minimum positive subnormal number is the smallest positive number representable in IEEE double format. The minimum positive normal number is often referred to as the underflow threshold. (The decimal values for the maximum and minimum normal and subnormal numbers are approximate; they are correct to the number of digits shown.)

Table: Bit Patterns in Double-Storage Format and their IEEE Values

Common Name	Bit Pattern (Hex)	Decimal Value
+ 0	00000000 00000000	0.0
- 0	80000000 00000000	-0.0
1	3ff00000 00000000	1.0
2	40000000 00000000	2.0
Max normal number	7fefffff ffffffff	1.7976931348623157e+308
Min positive normal number	00100000 00000000	2.2250738585072014e-308
Max subnormal number	000fffff ffffffff	2.2250738585072009e-308
Min positive subnormal number	00000000 00000001	4.9406564584124654e-324
+ ∞	7ff00000 00000000	Infinity
- ∞	fff00000 00000000	-Infinity
Not-a-Number	7ff80000 00000000	NaN

A NaN (Not a Number) can be represented by any of the many bit patterns that satisfy the definition of NaN. The hex values of the NaN shown in the above table is just one of the many bit patterns that can be used to represent a NaN.

Syntax diagram ***real***:

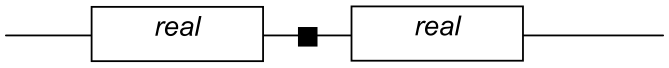


5.7.7 STRUCTURE OF COMPLEX NUMBERS (*COMPLEX*)

Complex numbers are handled as pairs of real numbers for which the syntax diagrams above apply. The number of items in a complex sequence is twice that of a real sequence. The order within the sequence is:

1st real part
1st imaginary part
2nd real part
2nd imaginary part
3rd real part ...

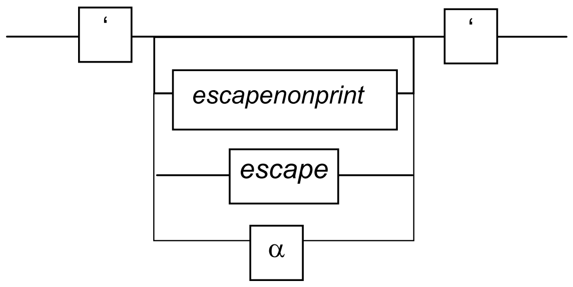
Syntax diagram ***complex***:



5.7.8 STRUCTURE OF CHARACTER CONSTANTS (*CHARACTER*)

Character constants are enclosed in single quotation marks and may contain only characters. For non-printable characters the usual C alternative representations are used (see also Escape sequences).

Syntax diagram ***character***:

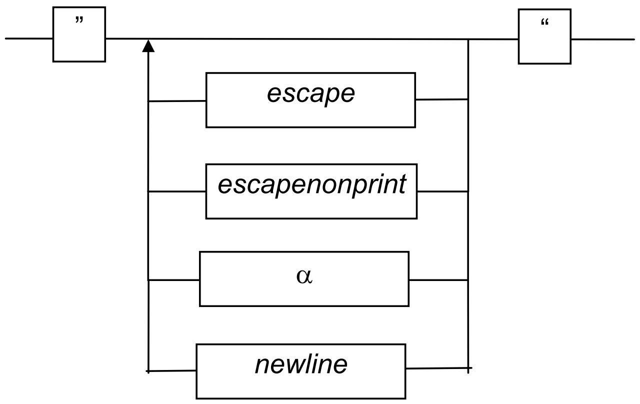


It is to be considered that character constants are only another representation of 8 bit integer values (*byte*). The internal representation is identical.

5.7.9 STRUCTURE OF STRINGS (*STRING*)

Strings are enclosed in double quotation marks and may contain any arbitrary number of representable characters. An empty string is also allowed. Composed text constants "..." + "..." are planned for future versions, but not yet implemented.

Syntax diagram ***string***:



Momentarily some Escape sequences are already used in networks. These sequences start with a backslash (\). In future versions new elements may be defined (e.g \a or \c). When using the backslash in strings it should be already written as „\\“, to be compatible with future versions (e.g. „\auto“ becomes „\\auto“).

Please note that a double quote inside a string MUST be preceded by a backslash, otherwise it would be regarded as end-of-string.

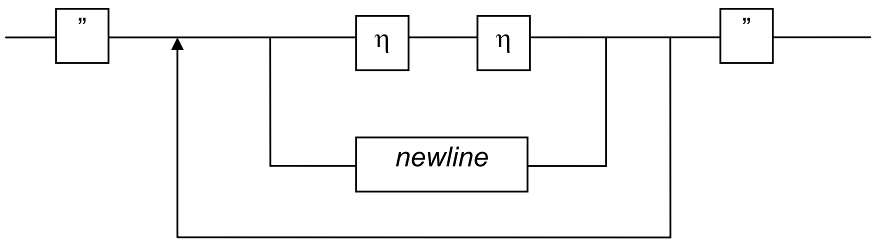
Please note that linefeeds that belong to the string must be encoded as \n, while .CR. and .LF. may be used to distribute a long string over several lines to enable easy editing and printing.

5.7.10 STRUCTURE OF BYTESTREAM (*BYTESTREAM*)

Bytestreams are represented like strings with special contents. Only the 16 Hex characters are allowed plus linefeeds to allow editing and printing of the ATF/CLA file.

Please note, that the length of the bytestream is NOT written. This eventually requires a recalculation of the bytestream length on reading.

Syntax diagram ***bytestream***:



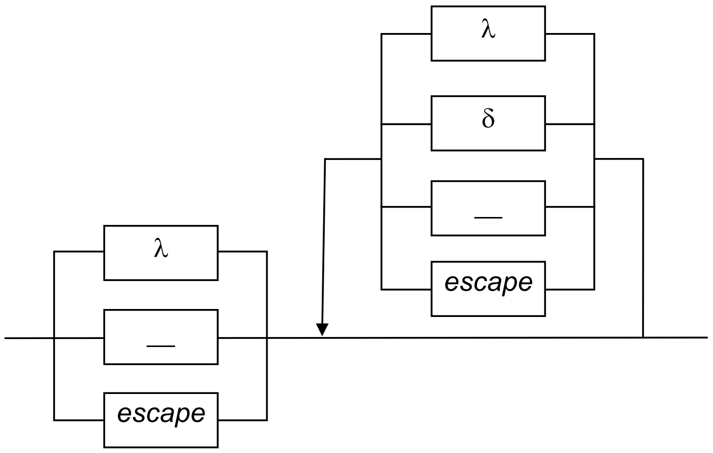
5.7.11 STRUCTURE OF IDENTIFIERS (*IDENTIFIER*)

For the definition of names of identifiers the rules of the ASAM standard apply. Special characters in identifiers which could lead to ambiguities within the syntax are started with a backslash.

Because the readability of an ATF/CLA file may be affected by identifiers with special characters, those kind of identifiers should probably not be used.

The names of identifiers must always start with a letter, an underscore or a defined Escape sequence. The length of names is not restricted in an ATF/CLA file.

Syntax diagram ***identifier***:



5.7.12 STRUCTURE OF FILENAMES (*FILENAME*)

The path- and filenames used as references in an ATF/CLA file must be transportable and interpretable on a wide range of computer architectures. Thus, a general format for the definition of path- and filenames is paramount, which has to be easily interpreted and converted for the target machine.

The Uniform Resource Locator (URL) used in the worldwide Internet is already working on different platforms. Through the naming conventions of the URL it will be possible for future ATF/CLA versions to have ATF/CLA files distributed within a network.

The URL describes a protocol for the target server, the target system the machine or server name, the port, the directory path and the filename. The URL has the following structure:

access_method://server_name[:port]/directory_path/filename

EXAMPLE:

The following example shows an URL for a resource on a remote machine (the master list of all World Wide Web servers).

With the protocol HTTP (Web) the server named `www.w3.org` is adressed. By using the standard port you find the directory `/hypertext/DataSources/WWW`, which contains the hypertext document `Geographical.html`.

```
http://www.w3.org/hypertext/DataSources/WWW/Geographical.html
```

Each file in the Internet is by means of the URL uniquely adressable.

If the protocol references the contents of a directory, the URL ends with a slash „/“. When referencing a file within the directory no slash follows.

The most used protocols in URLs are:

http:	The HyperText Transfer Protocol (the protocol of the World Wide Web (WWW))
file:	The URL contains only a filename, not an address.
ftp:	The File Transfer Protocol for transferring files.
gopher:	The Gopher protocol for transferring files and for navigation in Gopher systems
mailto:	The Internet Mail Protocol for sending E-mails to a recipient.
news:	The Internet-News protocol for polling articles or newsgroups.
rlogin:	A remote login for a defined machine.
telnet:	Starts a terminal session via Internet to a defined machine.

URLs as the ones described above are called absolute URLs, because they contain the full pathname of a file. A simplified format of URLs is called relative URL. This simplified format references other documents on the same server as the actual document. With relative URLs, an ATF/CLA file can reference directly the physical components without referencing the server. In the current version of ATF/CLA only relative URLs are supported.

Absolute URLs may reference any resource within the Internet, including local resources. Due to reasons explained later, relative URLs are better suited for local resources.

A relative URL implies the same access method, the same server name and the same directory path as the ATF/CLA file containing the URL. It describes the relative position of the component of the ATF/CLA file in relation to the actual ATF/CLA file.

EXAMPLE:

```
../../matrix_xyz.dat
```

Relative URLs make it possible to move the ATF/CLA directory structure to any position in the directory tree. A relative URL does not have to reference the directory of the actual document but may contain a reference relative to the root of the ATF/CLA file. This variant of relative URLs starts with a slash (/). It looks similar to an absolute URL but does not contain the access method and the server name.

The correct conversion of pathnames in both directions and the insertion of correct separators are the responsibility of the ATF/CLA reading and writing programs on the respective target systems.

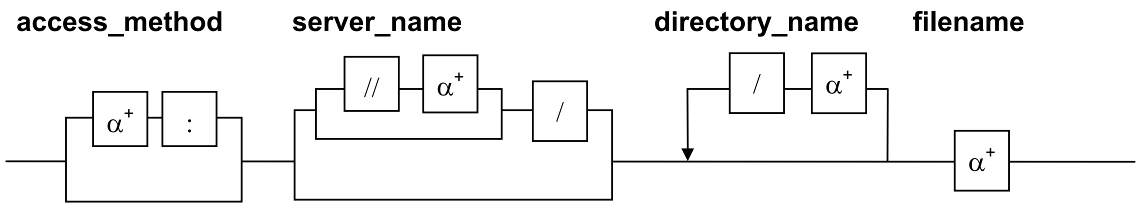
If filenames are too long for the target system (e.g. length of DOS-filenames 8.3), the filenames will be shortened following these rules: After the first colon within the filename a maximum of 3 more characters are allowed. None of these characters may be a colon. If there are more than 3 characters or another colon , the rest will be truncated. Unallowed characters are substituted by a tilde (swung dash).

The number of characters in front of the colon is counted and truncated to 8 characters. If this results in filenames which are not unique, the algorithms from Windows 95 can be used (truncating to 6 characters and using the latter 2 characters for numbering).

In this case manual editing of the ATF/CLA file is most likely. While all the needed files are concentrated within the FILES block the manual editing should not be too work-intensive.

To make the transfer of files to the world of UNIX easier filenames will always be written in lowercase, even on operating systems which are not case-sensitive (VAX/VMS, MS-Windows). Filenames in uppercase are only used on demand when path and filenames on the UNIX machine should be in uppercase.

Syntax diagram ***filename***:



5.7.13 PREDEFINED DATA TYPES (*DATATYPE*)

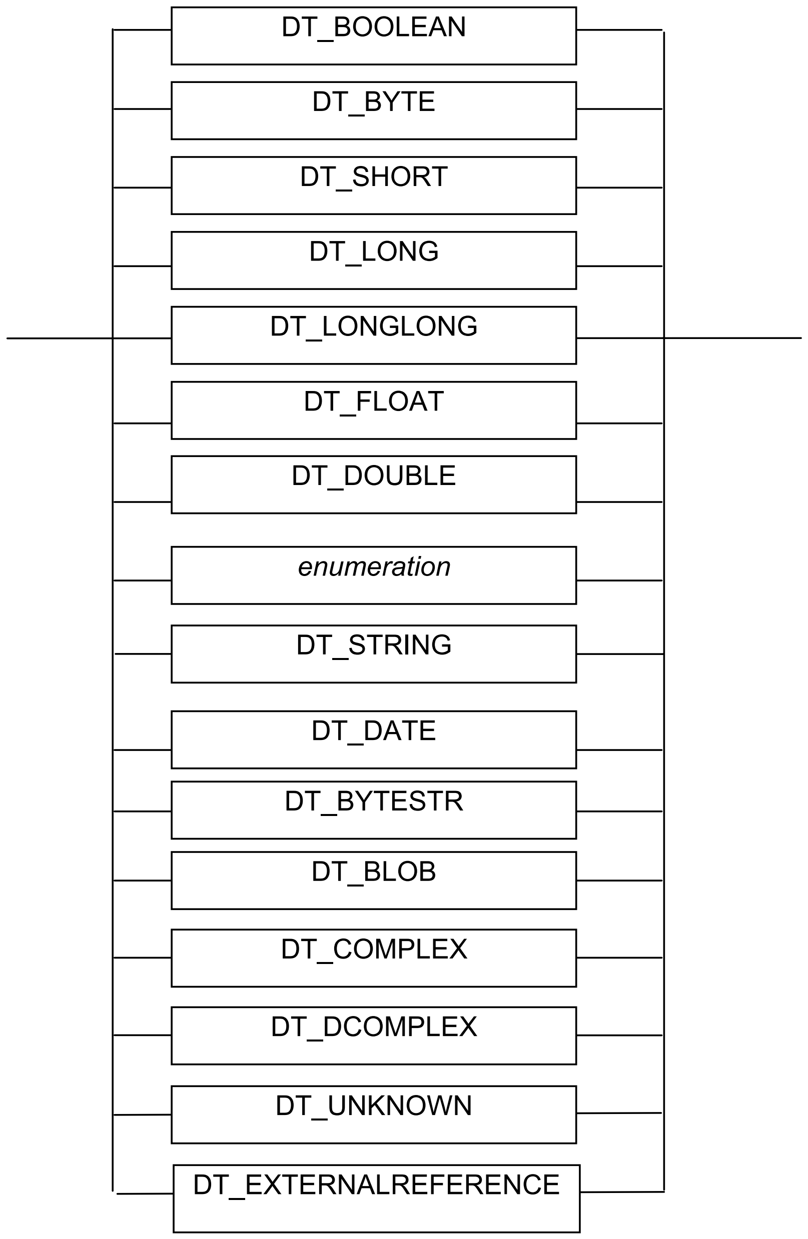
The data types of application and instance attributes may be freely chose DATATYPE n within the range of the following predefined data types:

DT_BOOLEAN	Logical value (TRUE/FALSE).
DT_BYTE	Single byte (8 Bit), no sign bit.
DT_SHORT	Integer with 16 Bit, including one sign bit.
DT_LONG	Integer with 32 Bit, including one sign bit.
DT_LONGLONG	Integer with 64 Bit, including one sign bit.
DT_FLOAT	Floating-point number with 32 bit.
DT_DOUBLE	Floating-point number with 64 bit.
DT_COMPLEX	Complex floating-point number with 32 bit parts.
DT_DCOMPLEX	Complex floating-point number with 64 bit parts.
DT_STRING	String enclosed in double quotation marks.
DT_ENUM	Enumeration denoting a limited set of strings that follows the keyword delimited by square brackets.
DT_BYTESTR	Byte stream: Binary Data
DT_BLOB	BinaryLargeObjectBlock: Named byte stream
DT_DATE	ASAM date type (string representation)
DT_UNKNOWN	Special type for baseattribute values from LocalColumn
DT_EXTERNALREFERENCE	External reference: three strings separated by colon: (description, mimetype, location)

The internal representation of these data types depend on the used machine type, the data storage system and the requirements of the users. Within the ATF/CLA file the data type DT_DATE is stored as normal string enclosed in quotation marks.

The conversion in a format applicable for databases or computations is the responsibility of the respective application.

Syntax diagram ***datatype***:



Please Note:

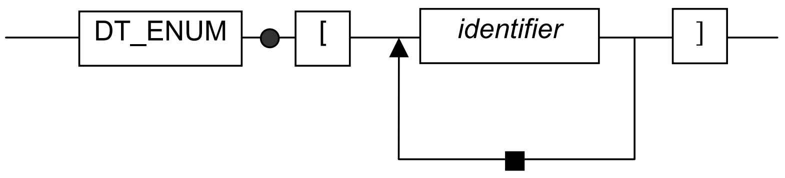
DT_UNKNOWN is only allowed at the base attribute “values” of AoLocalColumn.

For structure of enumeration see following section.

5.7.14 STRUCTURE OF ENUMERATION

The keyword DT_ENUM is used to describe an enumeration type in the application model. An enumeration has a limited set of allowed strings. This set is written behind the keyword DT_ENUM and is shall be enclosed in square brackets. The strings shall follow the rules for identifiers and they shall be separated by commas.

Syntax diagram ***enumeration***:



Within instance elements there are two possibilities to represent instantiated attributes which contain enumerations:

➢ Representation as String

The string must contain one of the identifiers which were defined between the brackets of the definition of APPLATTR.

EXAMPLE:

```
APPLELEM ...
APPLATTR exampleAttr, DATATYPE DT_ENUM [First, Second, Third];

...

INSTELEM ...
exampleAttr = "Third"
```

➢ Representation as Integer

The integer contains a null-based index to the list of strings which were defined between the brackets of the definition of APPLATTR.

EXAMPLE:

```
APPLELEM ...
APPLATTR exampleAttr, DATATYPE DT_ENUM [First, Second, Third];

...

INSTELEM ...
exampleAttr = 2; /*meaning: "Third"*/
```

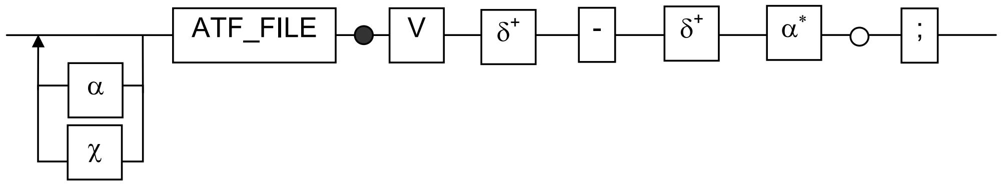
5.7.15 STRUCTURE OF THE ATF/CLA VERSION IDENTIFIER (*VERSION*)

The keyword ATF_FILE starts an ATF/CLA file and must be set at the beginning of the respective file. This keyword is followed by the version number of the ATF/CLA standard, on which the generation of the file is based. All information before this line are ignored, the interpretation starts after the recognition of the keyword „ATF_FILE". Reading programs must be able to ignore lines before the line with the keyword. Lines before this keyword are not stored.

Important comments should be inserted after the ATF_FILE line. This is due to applications which will not always copy lines before the ATF_FILE line.

When sending files via E-mail the routing information are inserted at the beginning of the respective file. In this case it is inconvenient for the recipient to erase these information with an editor.

Syntax diagram ***version***:



EXAMPLE:

```
This is the beginning of an ASAM file. All information
before the keyword ATF_FILE are ignored
/* even if they are marked as comment.*/

ATF_FILE V1.4.1;

// Created Fri May 09 16:28:45 2003
// by user GUEST on host ASAM with program APPLICATION1
```

The information when and where the ATF/CLA file has been created is an optional comment which has proven to be useful for tracking bugs and problems and for archive purposes.

5.7.16 STRUCTURE OF THE FILES BLOCK (*FILES*)

The FILES block is used to list all the physical files belonging to an ATF/CLA file. Such files can be either include files that are used by an INCLUDE statement (see next section) or binary component files that are used to represent large amounts of values like in local columns (see COMPONENT section). The ATF file (the one containing the "ATF_FILE" and "ATF_END;" statements) plus all the files listed in the FILES block can be regarded as one logical file. The FILES block is optional, because not every logical file is physically distributed over several files.

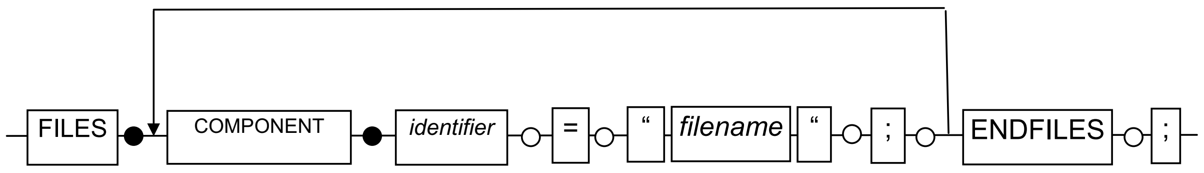
This block must be inserted before the first APPELEM-, INSTELEM instruction and must reference valid ASAM files. The order of the files is arbitrary. There is only one FILES block allowed within an ATF/CLA file. Therefore it must not be located in an included file.

A reading application must be sure that no files other than those defined in the Files block belong to the logical ATF/CLA file. All physical files belonging to a logical ATF/CLA file must be declared in the FILES block in COMPONENT statements. Please keep in mind that a file declared as such a component is not necessarily a binary component.

The definition of any number of components is allowed. The order of the components within the blocks is arbitrary. The indentations within the following example shall enhance the readability and are not mandatory.

The names of the components may then be used in INCLUDE instructions as well as in COMPONENTs in the data.

Syntax diagram ***files***:



The component definitions are only allowed within the FILES block. The filenames are:
- simple filenames (for files stored in the same directory as the ATF/CLA file);
- absolute pathnames (to be handled with care);
- relative pathnames (relative to the directory in which the ATF/CLA file is stored).

EXAMPLE:

```
FILES
   // Reference to more physical files
   // ... with the internal name "component_1"
   COMPONENT file_1 = "../data/k1.dat";
   COMPONENT file_2 = "atf_struct.inc";
   <...>
ENDFILES;
```

5.7.17 STRUCTURE OF THE INSTRUCTION INCLUDE (*INCLUDE*)

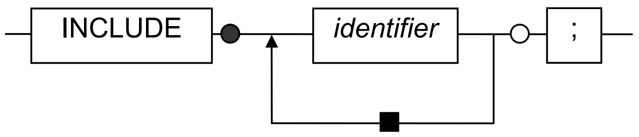
With the instruction INCLUDE a file is being inserted in the presently processed file. This instruction may be positioned anywhere in the file but not before the FILES section. The only condition is that the information contained in the Include file must be inserted syntactically correct at the respective position.

The INCLUDE instruction will be replaced immediately with the content of the referenced file. It is allowed to insert a file more than once in different positions of the main file.

Typically inserted files contain alias definitions, structure information or instance elements that are required in more than one ATF/CLA file and should only be defined once or data which have to be grouped. An example for such a file is a Measurement Quantity catalog which contains all the sizes and size groups required in a special environment.

Please note that these include files are ASCII files; their use is different from the use of binary component files.

Syntax diagram ***include***:



EXAMPLE:

```
FILES
     // Reference to more physical files
     // ... with the internal name "component_1"
     COMPONENT file_1 = "../data/k1.dat";
     COMPONENT file_2 = "atf_struct.inc";
     //...
ENDFILES;

INCLUDE file_1;
INCLUDE file_2, file_1;
```

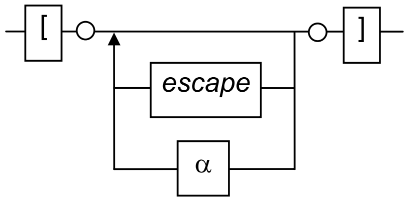
5.7.18 REFERENCES TO UNITS (*PHYS_UNIT*)

The physical unit of instance attributes may be defined by the information derived from the base element„unit“. It may also be described explicitly as explained below. In this case the name of the unit is written without quotation marks and enclosed in brackets.

EXAMPLE:

```
Power         = 123  [kW]
Acceleration  = 4.34 [m/s^2]
```

Syntax diagram ***phys_unit***:



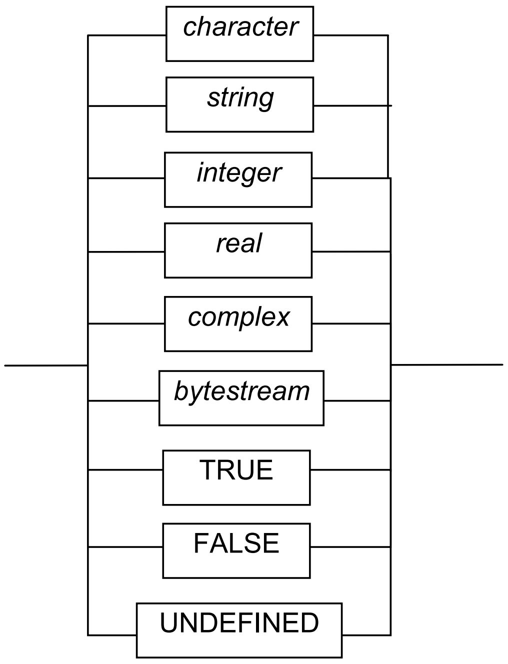
Please note: The unit string must not contain square brackets, they must be encoded as escapes.

In an ATF/CLA file such a definition may be set behind the attribute to assign the respective units.

5.7.19 VALUES OF DATA ATTRIBUTES (*DATAVALUE*)

The data values of attributes of the instances elements (see section „instance elements“) have the following format. This includes also the values for instance attributes. The attribute specification of the application elements (see section „application elements“) determine, which of the types is to be used.

Syntax diagram ***datavalue***:



5.7.20 STRUCTURE OF APPLICATION ELEMENTS (*APPLELEM*)

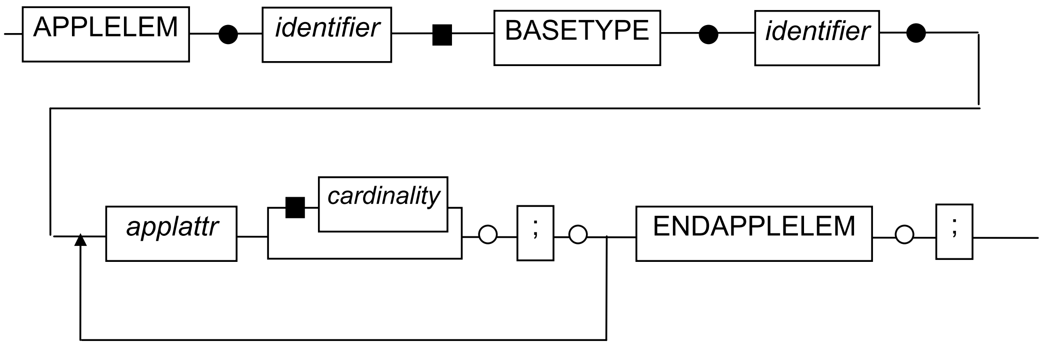
The APPLELEM construct allows to specify application elements that are derived from the standardized base elements. Therefore the keyword BASETYPE followed by a valid base element name is mandatory. The attributes may be derived from base attributes or application specific attributes may be specified. Some of the attributes may be references to other application elements. If the attribute is a reference, the application element which is referenced may be specified with the keyword REF_TO. If the attribute is not derived from a baseattribute the keyword REF_TO has to be given.

Attributes derived from base attributes inherit their data type from their base attribute - thus specifying a data type for derived attributes is not necessary. For each attribute a cardinality may be specified.

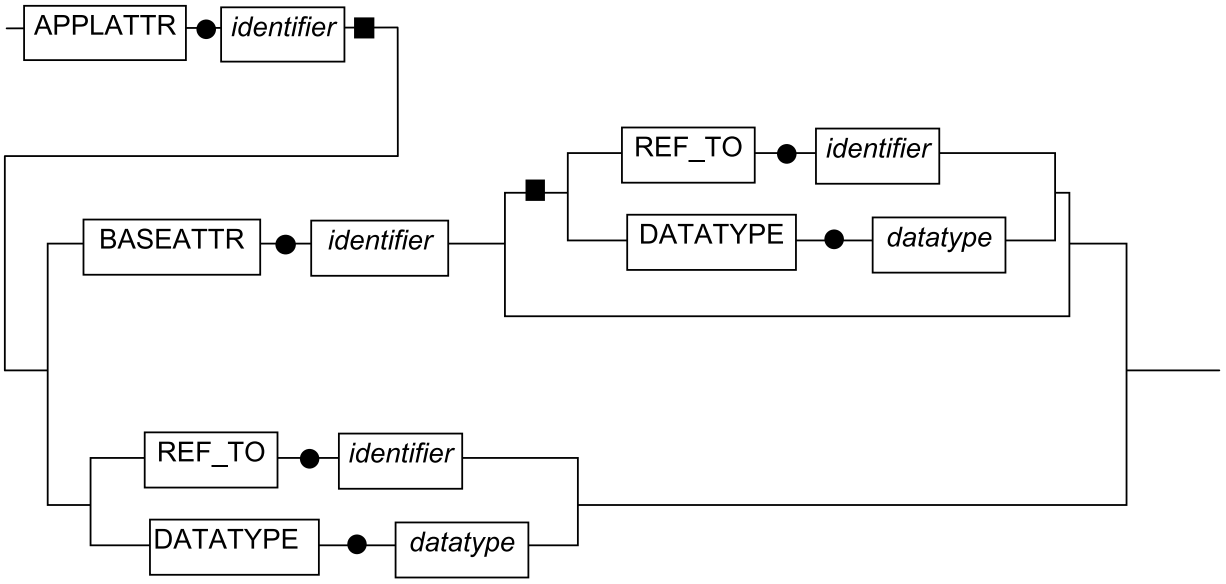
The keyword CARDINALITY specifies how many instances can be referenced or how many datavalues may be given. The keyword is followed by two integers. The first integer defines the minimum number of values and the second integer the maximum number. If the maximum number is unknown (or infinit) the second integer is replaced by the keyword MANY. If the CARDINALITY clause is not present a default of 0 (minimum number) and 1 (maximum number) is assumed.

The definition of any number of application attributes is allowed. The order of the definitions within the blocks is arbitrary. The indentations within the following example shall enhance the readability and are not mandatory.

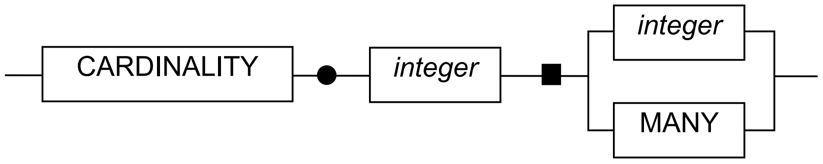
Syntax diagram ***applelem***:



Syntax diagram ***applattr***:



Syntax diagram ***cardinality***:



Within the ASAM data model application elements are related to other application elements. For example a test may posses several measurements and a measurement can be possessed by a test.

In an ATF/CLA file such relation attributes are declared with the keyword REF_TO. The entity „measurement“ contains for example the attribute „BASEATTR measurement_quantities“.

EXAMPLE:

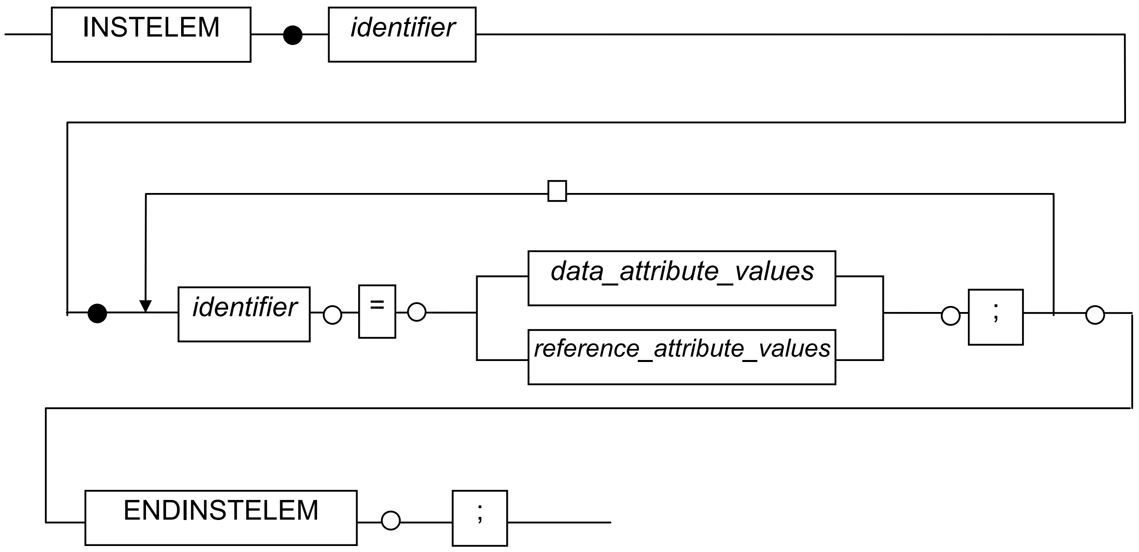
```
APPLELEM SpecialMeasurement, BASETYPE AoMeasurement
   APPLATTR ID, BASEATTR id;
   APPLATTR Name, BASEATTR name;
   APPLATTR myMeasurementQuantities,
        BASEATTR measurement_quantities,
        REF_TO MeasurementQuantity,
        CARDINALITY 0,MANY;
ENDAPPLELEM;

APPLELEM MeasurementQuantity, BASETYPE AoMeasurementQuantity
   APPLATTR ID, BASEATTR id;
   APPLATTR Name, BASEATTR name;
   //...
ENDAPPLELEM;
```

5.7.21 STRUCTURE OF INSTANCE ELEMENTS (*INSTELEM*)

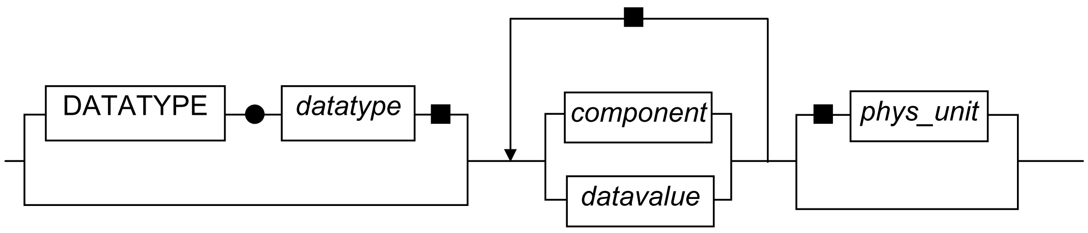
The instruction instelem is used to define instance elements. Within the ASAM data model instance elements relate to other instance elements. For example a test may have several measurements, on the other hand a measurement may tell to which test it belongs.

The order of attribute assignments is arbitrary. Application attributes without assigned values have the default value UNDEFINED. Within an instance element an arbitrary number of new instance attributes may be defined and assigned. For these instance attributes a data type has to be given.

Syntax diagram ***instelem***:



Syntax diagram ***data_attribute_values***:



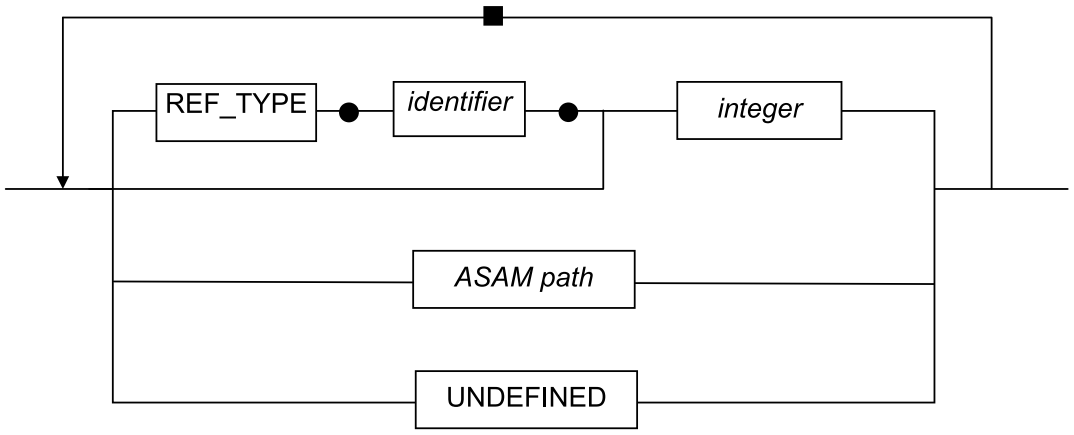
Please remember:
A circle represents a whitespace, a square represents a separator;
hollow means optional, full means mandatory.

Special attribute „values“ of AoLocalColumn

The base element AoLocalColumn holds in its attribute „values“ a sequence of data of a certain type. This sequence shall NOT appear as a separate object, therefore no reference to a sequence is given in the attribute „values“. Instead in this attribute are written DIRECTLY the values contained in the sequence. The type of the sequence must be indicated by the construct „DATATYPE *datatype*“ with *datatype* = DT_...

Please note this handling in the example given in section 5.7.22 (example 1).

Syntax diagram ***reference attribute values***:



EXAMPLE:

```
INSTELEM measurement
   //<...>
   ID = 47
   Name = "Example for a measurement";
   //<...>
ENDINSTELEM;
```

For identifying an instance the attribute which is derived from the baseattribute id will be taken. This ID is unique within the instances of an application element and can be used to reference the respective instances. The IDs in the referencing element are separated by commas.

EXAMPLE:

```
INSTELEM Measurement
   ID = 121;
   Name = "Exhaust gas testing";
   //...
   Measurement_Quantitys = 200, 201, 202;
ENDINSTELEM;

INSTELEM Measurement_Quantity
   ID = 200;
   Name = "Time";
   //...
ENDINSTELEM;
```

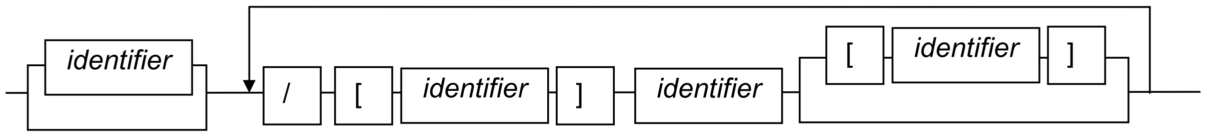
Incomplete data on ATF/CLA-File:

If not all references can be satisfied in an ATF/CLA-File, then for some Ids it is not possible to give the integer value. In this case it may be replaced by a string that describes the location of the reference by means of the „ASAM Path“.

An ASAM Path has an optional Service-Name (if included: „Full ASAM Path“, if omitted: „Local ASAM Path“) and then an arbitrary number of instance specifications, each of them containing the application element name, the instance element name and optional the version. Name and version are the content of the string values of the attributes „name“ resp. „version“.

In the following syntax diagram these 4 names are represented by identifiers.

Syntax diagram ***ASAM path***:



In the first Identifier (service name) may be given a file name, on which the instance can be found.

EXAMPLE:	DETAILED REFERENCE SPECIFICATION:
	The following example shows the use of the keyword REF_TYPE for the case that an application element was subtyped further and instance ids are not unique within all subtypes. In the example there are the entities Eq1 and Eq2 which are both subtypes of the application element Equipment (which may be a subtype of the base element AoTestEquipmentPart). If the application model specifies a reference to „Equipment“, then the construct shown here can distinguish the instances of Eq1 and Eq2.

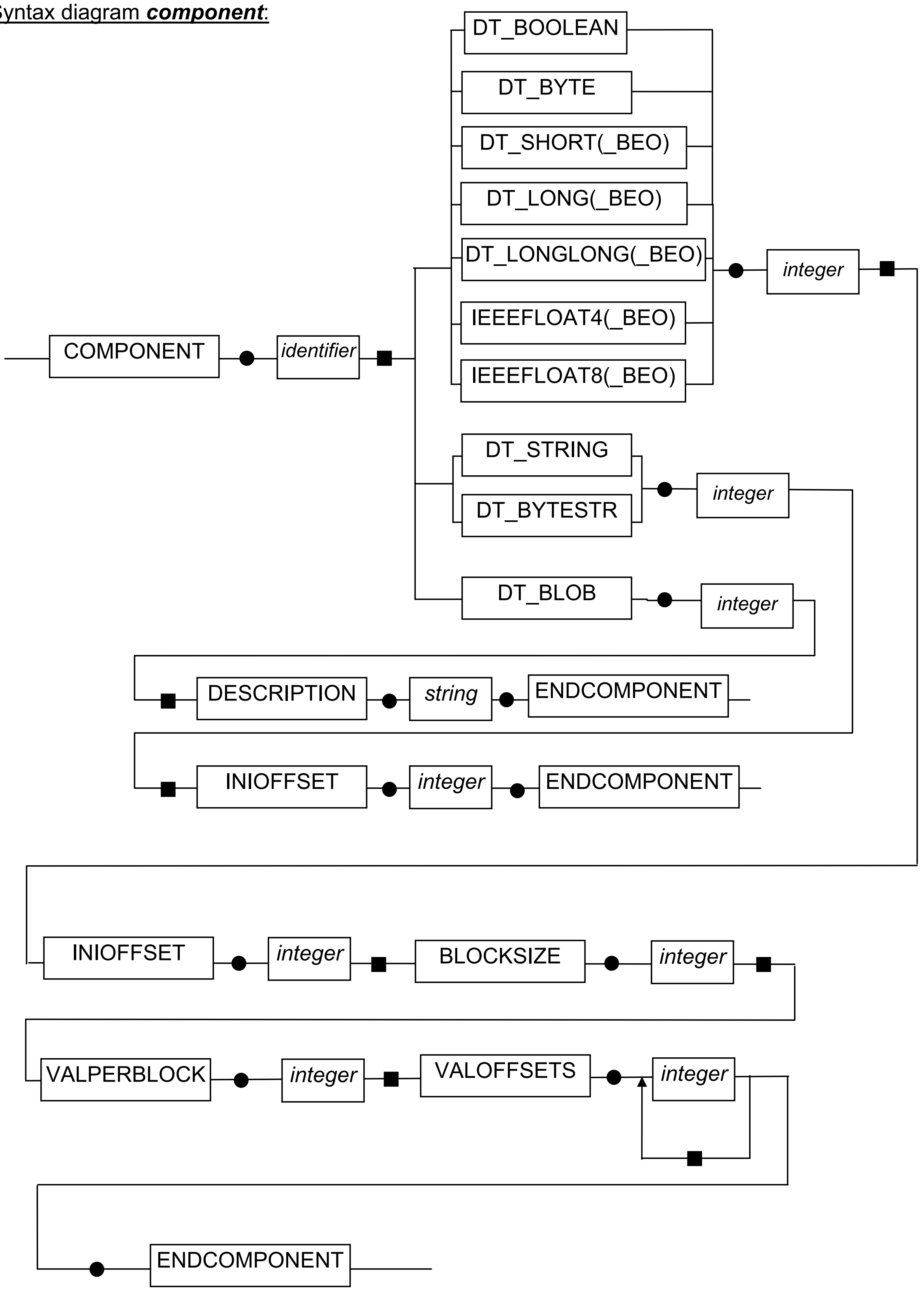
```
INSTELEM ThisEq1
    ID = 1;
    Name = "Engines";
    //...
    Tests = REF_TYPE Eq1 200, 201,
             REF_TYPE Eq2 200;
ENDINSTELEM;

INSTELEM Eq1
    ID = 200;
    //...
ENDINSTELEM;

INSTELEM Eq1
    ID = 201;
    //...
ENDINSTELEM;

INSTELEM Eq2
ID = 200;
//...
ENDINSTELEM;
```

5.7.22 STRUCTURE OF COMPONENTS (*COMPONENT*)

In ASAM ODS the measured data is described by the base element *Local Column*. These columns are referred by measurement quantities and submatrices. While other elements like Measurement_quantity and Submatrix can be dealt with as normal base elements, the local column often refers large external binary datasets. Mainly for this reason, the COMPONENT construct was provided in this ATF/CLA specification. This construct allows to include binary data in an ASCII file and therefore can save considerable time and space in transmission and storage.

Each component must have been defined in advance within the FILES block. The specification for the binary data must be added where the binary component has to be inserted, i.e. usually after the equals sign of an attribute in an instance element.

Syntax diagram ***component***:



The whole component description has to be inserted in the attribute instead of the ASCII representation of the values. The format is given in the syntax diagram in this section, any additional information shall be given here.

Component identification:

Behind the keyword COMPONENT an identifying name must be given. This name must have been announced in the FILES block at the beginning of the ATF/CLA file. Please note that in case of a blob this name is not identical with the blob header (which is put after the DESCRIPTION keyword).

Type specification:

The physical data type specifies the manner in which the data is stored on the file. In this specification simple integer types, the IEEE-floating point formats and several byte formats are accepted:

DT_BOOLEAN
DT_BYTE
DT_SHORT
DT_LONG
DT_LONGLONG
IEEEFLOAT4
IEEEFLOAT8
DT_SHORT_BEO
DT_LONG_BEO
DT_LONGLONG_BEO
IEEEFLOAT4_BEO
IEEEFLOAT8_BEO
DT_STRING
DT_BYTESTR
DT_BLOB

Length specification:

The integer value behind the type specifies the amount of data to be read from the component. In case of DT_STRING, DT_BYTESTR and DT_BLOB it denotes the number of bytes to be read; in all other cases (integers, floats, bits and bytes) it denotes the number of values to be read. In the case of blobs this specification is equivalent with the length of the file on which the blob is located, because in a COMPONENT only one blob can be referred. All other types may occur in sequences on a component file.

Representation of BOOLEAN:

In a binary component file each boolean shall be represented by one bit (0=false, 1=true). These bits are filled into a sequence of bytes. If the number of bits is not sufficient to fill a byte completely, then the rest of that byte shall remain unused. The byte is filled from left to right, i.e. beginning with the most significant bit. In the byte sequence (by_1, by_2, by_3,...) the n-th boolean (n is one-based) can be located using the following integer formula. Assuming the bit positions in a byte numbered (again from left to right) 7,6,5,4,3,2,1,0 then the n-th boolean is in the lby-th byte at bit position lbi:

lby = (n+7) / 8

lbi = lby*8 - n

The number of used bytes (Nby) can be calculated from the number of booleans (Nbi) by integer arithmetic

Nby = 1 + (Nbi-1) / 8

In the syntax diagram the integer after DT_BOOLEAN shall denote the number of bits (i.e. the number of given booleans.

Representation of BYTE:

This data type is represented simply by bytes. Thus the integer behind DT_BYTE denotes the number of byte values in the sequence, in case of a single value it is set to 1.

Representation of SHORT:

This 16-bit integer data type is represented by 2 bytes. If the keyword DT_SHORT is given, then the least significant byte comes first (lower address), the most significant byte comes last (higher address). If the keyword DT_SHORT_BEO is given, then the most significant byte comes first, the least significant byte comes last; this order is also known as "Big Endian Order". The integer behind DT_SHORT (resp. DT_SHORT_BEO) denotes the number of integer values in the sequence, in case of a single value it is set to 1.

Representation of LONG:

This 32-bit integer data type is represented by 4 bytes. If the keyword DT_LONG is given, then the least significant byte comes first (lower address), the most significant byte comes last (higher address). If the keyword DT_LONG_BEO is given, then the most significant byte comes first, the least significant byte comes last; this order is also known as "Big Endian Order". The integer behind DT_LONG denotes the number of integer values in the sequence, in case of a single value it is set to 1.

Representation of LONGLONG:

This 64-bit integer data type is represented by 8 bytes. If the keyword DT_LONGLONG is given, then the least significant byte comes first (lower address), the most significant byte comes last (higher address). If the keyword DT_LONGLONG_BEO is given, then the most significant byte comes first, the least significant byte comes last; this order is also known as "Big Endian Order". The integer behind DT_LONGLONG denotes the number of integer values in the sequence, in case of a single value it is set to 1.

Representation of FLOAT:

This data type is represented by 4 bytes. If the keyword IEEEFLOAT4 is given, then the least significant byte comes first (lower address), the most significant byte comes last (higher address). If the keyword IEEEFLOAT4_BEO is given, then the most significant byte comes first, the least significant byte comes last; this order is also known as "Big Endian Order". The location of sign, exponent and mantissa is described in the section “Structure of Floating Point Numbers”. The integer behind IEEEFLOAT4 denotes the number of float values in the sequence, in case of a single value it is set to 1.

Representation of DOUBLE:

This data type is represented by 8 bytes. If the keyword IEEEFLOAT8 is given, then the least significant byte comes first (lower address), the most significant byte comes last (higher address). If the keyword IEEEFLOAT8 _BEO is given, then the most significant byte comes first, the least significant byte comes last; this order is also known as "Big Endian Order". The location of sign, exponent and mantissa is described in the section “Structure of Floating Point Numbers”. The integer behind IEEEFLOAT8 denotes the number of float values in the sequence, in case of a single value it is set to 1.

Representation of COMPLEX:

Complex numbers are controlled by the keywords IEEEFLOAT4 and IEEEFLOAT8 (resp. IEEEFLOAT4_BEO and IEEEFLOAT8_BEO). The number of values must be doubled (i.e. only even numbers are legal). For a single complex value the integer after the data type is 2, for a sequence of 7 complex values the integer is 14. The order in complex sequences is

- real part of first sequence member
- imaginary part of first sequence member
- real part of second sequence member
- imaginary part of second sequence member
- ...

Representation of STRING:

This data type is represented by bytes. Each string is terminated by a NULL byte which increases the required space for that string by 1. In a sequence of strings each member of the sequence has a terminating NULL byte, and these NULL bytes are included in the total length of the component which is given in the integer after DT_STRING. The data type DT_STRING may not be mixed with other data types on the same component file.

Representation of BYTESTR:

This data type is represented by bytes. In a bytestream the first 4 bytes denote the number of bytes following after these 4 bytes. In a sequence of bytestreams the next length indication follows immediately after the last byte of the previous member of the sequence. Please note that the 4 length indication bytes are given in “Big Endian Order”, i.e. with the most significant byte first.

EXAMPLE:

2 bytestreams with contents "ABCDEFG" and "XYZ" shall be represented by
00 00 00 07 41 42 43 44 45 46 47 00 00 00 03 58 59 5A
The total length is 18, this 18 is written in the integer after DT_BYTESTR.

The data type DT_BYTESTR may not be mixed with other data types on the same component file.

Representation of BLOB:

A blob is a stream of bytes with a known length and a header. While in ASCII format such a blob is represented by two strings (the first one contains the header, the second one contains the bytestream), this is handled different in components:

- The header of the blob shall go into the string given after the keyword DESCRIPTION;
- the length of the blob is given in the integer after DT_BLOB (this is compatible with the other length indications since there is always only one blob at a time);
- the content of the blob is on the binary file (which contains nothing but this blob).

The data type DT_BLOB may not be mixed with other data types on the same component file.

Other parameters:

With INIOFFSET the number of bytes in the file header is specified. This file header is skipped when accessing the file. Since a blob always covers a whole file, there is no INIOFFSET in blob components.

The rest of the file (beginning after the header) is regarded as one ore many blocks. A block consists of values. These values may belong to different components and (in case of numerical data) may have different data types. Within a block also several values may belong to the same component, see examples at the end of this section. For all numerical data types a blocksize must be specified. The BLOCKSIZE is given in Byte.

The number of values in each block belonging to the desired component is given with VALPERBLOCK. For each of these values the offset within a block must be specified in bytes. All these offsets are collected behind the keyword VALOFFSETS. The procedure is shown in the examples below.

EXAMPLE 1: **LOCAL COLUMN WITHOUT COMPONENT**

In the following instance the data is physically given on the ATF/CLA file:

```
INSTELEM localcolumn
   ID = 2;
   Values = DATATYPE DT_DOUBLE
   ,260.
   ,270.
   ,280.
   ,290.
   ,300.
   ,310.
   ,320.
   ,330.
   ,340.
   ,350.
   ,360.;
ENDINSTELEM;
```

EXAMPLE 2: **ONE LOCAL COLUMN ON ONE FILE**

This file contains one measurement quantity with one measured value channel with double precision floating-point numbers.

```
INSTELEM localcolumn
   ID = 3;
   Values = DATATYPE DT_DOUBLE,
   COMPONENT binary_file,IEEEREAL8 11, INIOFFSET 0,
      BLOCKSIZE 8, VALPERBLOCK 1, VALOFFSETS 0
   ENDCOMPONENT;
ENDINSTELEM;
```

EXAMPLE 3: **TWO LOCAL COLUMNS ONE AFTER ANOTHER**

This file contains two measurement quantities with

- one measured value channel with 10 double precision floating-point numbers.
- one measured value channel with 10 short integer numbers.

The second COMPONENT has an offset of 800 bytes to skip the first COMPONENT.

```
INSTELEM localcolumn
   ID = 4;
   Values = DATATYPE DT_DOUBLE,
   COMPONENT binary_file,IEEEREAL8 10, INIOFFSET 0,
      BLOCKSIZE 8, VALPERBLOCK 1, VALOFFSETS 0
   ENDCOMPONENT;
ENDINSTELEM;
INSTELEM localcolumn
    ID = 3;
    Values = DATATYPE DT_SHORT,
    COMPONENT binary_file, DT_SHORT 10, INIOFFSET 800,
        BLOCKSIZE 8, VALPERBLOCK 1, VALOFFSETS 0
    ENDCOMPONENT;
ENDINSTELEM;
```

EXAMPLE 4: **TWO LOCAL COLUMNS ALTERNATING ON ONE FILE**

This file contains two measurement quantities which were measured with the same sampling rate.

- one generated time channel with double precision floating-point numbers
- one measured value channel with short integer numbers.

These values were written alternately to the file. They are referred by two local columns.

```
INSTELEM localcolumn
   ID = 6;
   Values = DATATYPE DT_DOUBLE,
   COMPONENT binary_file,IEEEREAL8 1000, INIOFFSET 0,
      BLOCKSIZE 10, VALPERBLOCK 1, VALOFFSETS 0
   ENDCOMPONENT;
ENDINSTELEM;

INSTELEM aolocalcolumn
   ID = 7;
   Values = DATATYPE DT_SHORT,
   COMPONENT binary_file, INT2 1000, INIOFFSET 0,
      BLOCKSIZE 10, VALPERBLOCK 1, VALOFFSETS 8
   ENDCOMPONENT;
ENDINSTELEM;
```

EXAMPLE 5: **THREE LOCAL COLUMNS WITH DIFFERENT SAMPLING RATES ON ONE FILE**

This file contains three measurement quantities measured with different sampling rates:

- one value channel with double precision floating-point numbers with 10 Hertz
- one value channel with short integer numbers with 20 Hertz
- one value channel with long integer numbers with 30 Hertz.

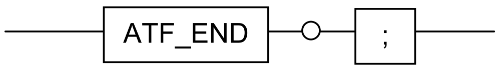
The values were written to the file as they came from the measuring instrument. Before the first measured data the instrument sents 32 Bytes of parameters.
Three local columns are needed.

```
INSTELEM localcolumn
   ID = 8;
   Values = DATATYPE DT_DOUBLE,
   COMPONENT binary_file,IEEEREAL8 10000, INIOFFSET 32,
      BLOCKSIZE 24, VALPERBLOCK 1, VALOFFSETS 0
   ENDCOMPONENT;
ENDINSTELEM;

INSTELEM localcolumn
   ID = 9;
   Values = DATATYPE DT_SHORT,
   COMPONENT binary_file,INT2 20000, INIOFFSET 32,
      BLOCKSIZE 24, VALPERBLOCK 2, VALOFFSETS 8,18
   ENDCOMPONENT;
ENDINSTELEM;

INSTELEM localcolumn
   ID = 10;
   Values = DATATYPE DT_LONG,
   COMPONENT binary_file,INT4 30000, INIOFFSET 32,
      BLOCKSIZE 24, VALPERBLOCK 3, VALOFFSETS 10,14,20
   ENDCOMPONENT;
ENDINSTELEM;
```

5.7.23 THE STRUCTURE OF ENDFILE (*ENDFILE*)

The ATF/CLA-File must be ended with the Keyword ATF_END.

Syntax diagram ***endfile***:



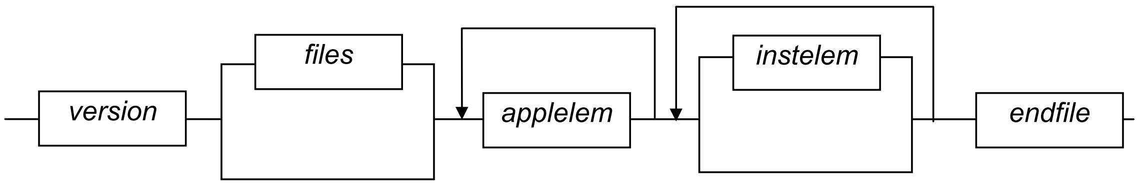
5.8 OVERVIEW OVER THE STRUCTURE OF AN ATF/CLA FILE (*ATF_OVERVIEW*)

This section gives an overview over the structure of an ATF/CLA file referring to the above mentioned elements. The syntax diagram only contains the main components. Syntax elements like include which can be placed anywhere in the diagram are not shown to keep the diagram readable.

It is to be considered that no one of the following elements except the ATF/CLA file identifier has to be available. It may be possible for an ATF/CLA file to be valid without the definition of an own environment, if the reading application integrates the information from the file in an already existing environment.

As already mentioned before, the logically sensible assignment of information to already existing data structures is the responsibility of the application programs. The ATF/CLA file is only to be seen as a standardized communication platform similar to a natural language. The understanding of complex relationships defined in this language requires the respective knowledge from all communication partners.

Syntax diagram ***atf_overview***:



5.9 SECURITY INFORMATION ON ATF/CLA FILES

5.9.1 THE USE OF SECURITY INFORMATION ON ATF/CLA

Security information is used to accompany archive data in order to be able to restore the data completely, i.e. together with the access rights. For exchange purposes it is necessary to write ATF/CLA files without security information. ATF/CLA writers therefore need a switch to select either “with” or “without” security information. If “with” is selected then several groups of security information can be written. These may include or exclude:

- users and user groups
- ACLs
- security level
- initial rights (ACL templates).

ATF/CLA readers need at least a defined default behavior if they encounter a file with security information; this behavior is either “use ” or “ignore” security information, possibly also “use partly” for which the part to be used must be specified in a similar way as on writing.

- ➢ On export the writing of security information on ATF/CLA file is optional.
- ➢ On import the use of security information from the ATF/CLA file is optional.

On reading a file without security information the local security information is not influenced.

5.9.2 WRITING SECURITY INFORMATION ON ATF/CLA

Security information consists of instances of entities that are subtypes of

- ➢ AoUser (such subtypes are application elements and must be described in the metadata section)
- ➢ AoUsergroup (such subtypes are application elements and must be described in the metadata section)
- ➢ ACL (these are: ACLA, ACLI, ACLTemplate)

and instances of the following two entities:

- ➢ SecurityLevel
- ➢ InitialRightsAt

Since the subtypes of AoUser and AoUsergroup are application elements they are mandatorily defined in the metadata section of the ATF/CLA file. The subtypes of ACL are not derived by the user, only the 3 predefined subtypes

- ACLA (used for an application element and -if an attribute name is specified- for the attribute of an application element)
- ACLI (used for instance elements)
- ACLTemplate (used for initial rights)

are allowed.

SecurityLevel, InitialRightsAt and the ACLs are **not** application elements, instead these entities belong only to the base model. Therefore they are **not** described in the metadata (*applelem*) section of the ATF/CLA file.

Please note that the attribute “users” in the ACLs refers to an application element derived from AoUserGroup. Therefore the instantiated attribute must use the REF_TYPE construct to specify the application element to which the ID in the attribute value points.

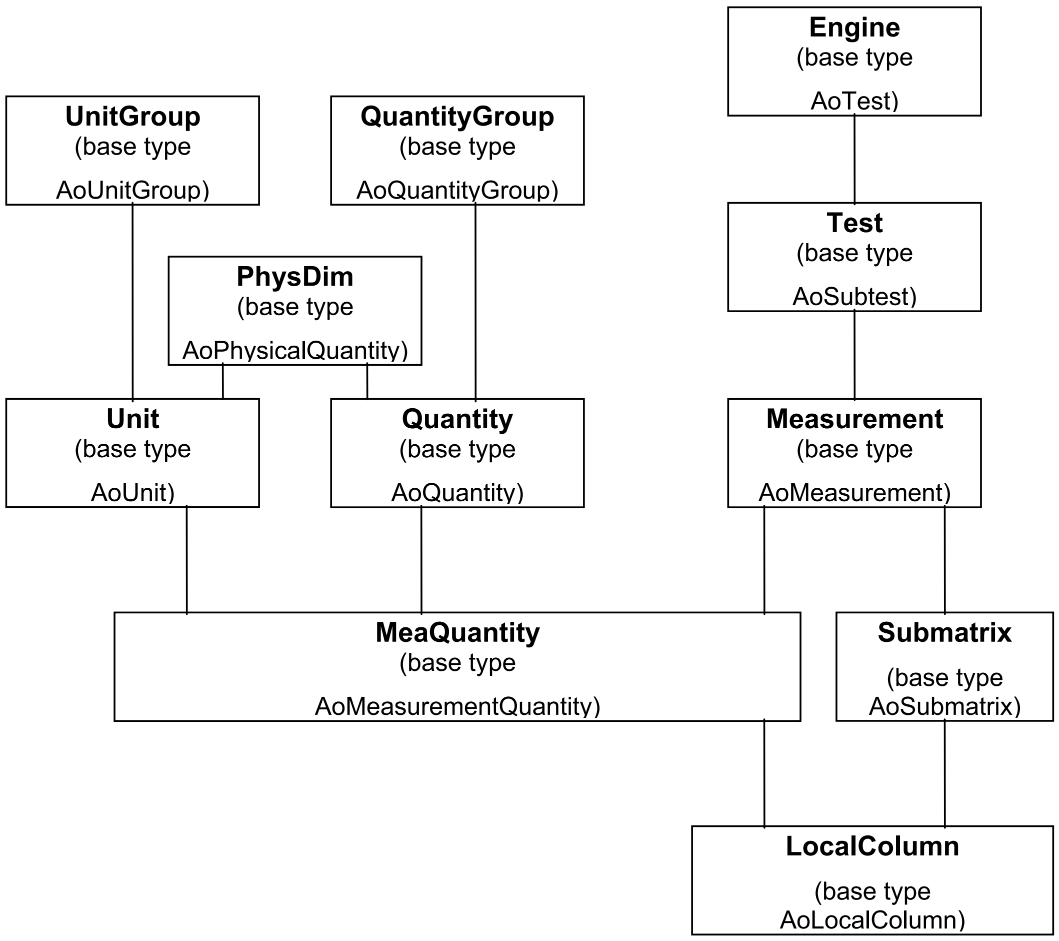
No special syntax is applied to the security information. It is written according to the same rules as all other data.

- The entities for users and user groups are written in the metadata section together with the other application elements.
- The instances are written in the data section together with the other instance elements.

The example file at the end of this chapter contains security information and -in comments- an explanation how it shall be interpreted.

5.10 EXAMPLE FOR AN APPLICATION

The following example shows a short part of an ATF/CLA file. Main elements of the model are shown in the following figure. For detailed information on the base model please refer to chapter **ASAM ODS Base Model**.


Engine (base type AoTest)
UnitGroup (base type AoUnitGroup)
QuantityGroup (base type AoQuantityGroup)
Test (base type AoSubtest)
PhysDim (base type AoPhysicalQuantity)
Unit (base type AoUnit)
Quantity (base type AoQuantity)
Measurement (base type AoMeasurement)
MeaQuantity (base type AoMeasurementQuantity)
Submatrix (base type AoSubmatrix)
LocalColumn (base type AoLocalColumn)

```
ATF_FILE V1.4.1;
/*
//---------------------------------------------------------------------------
// EXAMPLE 3 --- 22.9.2000
//---------------------------------------------------------------------------
//***************************************************************************
//
// Application Structure
//
//***************************************************************************
*/
/*
//---------------------------------------------------------------------------
//
// Hierarchic ordering of tests and measurements
// Application elements
//             Engine, Test,      Measurement,   MeaQuantity
// of base type AoTest, AoSubtest, AoMeasurement, AoMeasurementQuantity
//
// Security information is included
//
//---------------------------------------------------------------------------
*/

APPLELEM Engine, BASETYPE AoTest
  APPLATTR EngineName,        BASEATTR name,          DATATYPE DT_STRING;
  APPLATTR EngineId,          BASEATTR id,            DATATYPE DT_LONGLONG;
  APPLATTR EngineVersion,     BASEATTR version,       DATATYPE DT_STRING;
  APPLATTR EngineDescription, BASEATTR description,   DATATYPE DT_STRING;
  APPLATTR version_date,      BASEATTR version_date,  DATATYPE DT_DATE;
  APPLATTR CreateDate,                                DATATYPE DT_DATE;
  APPLATTR EngineType,                                DATATYPE DT_STRING;
  APPLATTR bore,                                      DATATYPE DT_FLOAT;
  APPLATTR cylindernumber,                            DATATYPE DT_SHORT;
  APPLATTR stroke,                                    DATATYPE DT_FLOAT;

  APPLATTR SubTests,          BASEATTR children,      REF_TO Test,
                                                      CARDINALITY 0,MANY;
  APPLATTR EngineUser,                                REF_TO User;
ENDAPPLELEM;


APPLELEM Test, BASETYPE AoSubTest
  APPLATTR TestName,         BASEATTR name,           DATATYPE DT_STRING;
  APPLATTR TestId,            BASEATTR id,              DATATYPE DT_LONGLONG;
  APPLATTR TestVersion,       BASEATTR version,         DATATYPE DT_STRING;
  APPLATTR TestDescription,   BASEATTR description,     DATATYPE DT_STRING;
  APPLATTR version_date,      BASEATTR version_date,    DATATYPE DT_DATE;
  APPLATTR CreateDate,                                  DATATYPE DT_DATE;
  APPLATTR TestType,                                    DATATYPE DT_STRING;
  APPLATTR TestComment,                                 DATATYPE DT_STRING;
  APPLATTR exhaust,                                     DATATYPE DT_STRING;
  APPLATTR air_filter,                                  DATATYPE DT_STRING;

  APPLATTR Measurements,      BASEATTR children,        REF_TO Measurement,
                                                        CARDINALITY 0,MANY;
  APPLATTR MainTest,          BASEATTR parent_test,     REF_TO Engine;
  APPLATTR TestUser,                                    REF_TO User;
ENDAPPLELEM;


APPLELEM Measurement, BASETYPE AoMeasurement
  APPLATTR MeaName,           BASEATTR name,            DATATYPE DT_STRING;
  APPLATTR MeaId,             BASEATTR id,              DATATYPE DT_LONGLONG;
  APPLATTR MeaVersion,        BASEATTR version,         DATATYPE DT_STRING;
  APPLATTR MeaDescription,    BASEATTR description,     DATATYPE DT_STRING;
  APPLATTR version_date,      BASEATTR version_date,      DATATYPE DT_DATE;
  APPLATTR MeasurementBegin, BASEATTR measurement_begin,DATATYPE DT_DATE;
  APPLATTR MeasurementEnd,    BASEATTR measurement_end,  DATATYPE DT_DATE;
  APPLATTR CreateDate,                                    DATATYPE DT_DATE;
  APPLATTR MeaType,                                     DATATYPE DT_STRING;
  APPLATTR MeaComment,                                  DATATYPE DT_STRING;
  APPLATTR ROZ,                                         DATATYPE DT_FLOAT;

  APPLATTR Test,              BASEATTR test,            REF_TO Test;
  APPLATTR MeaQuantities,     BASEATTR measurement_quantities,
                                     REF_TO MeaQuantity, CARDINALITY 0,MANY;
  APPLATTR Submatrices,       BASEATTR submatrices,
                                     REF_TO Submatrix, CARDINALITY 0,MANY;
ENDAPPLELEM;


APPLELEM MeaQuantity, BASETYPE AoMeasurementQuantity
  APPLATTR MeaQName,          BASEATTR name,            DATATYPE DT_STRING;
  APPLATTR MeaQId,            BASEATTR id,              DATATYPE DT_LONGLONG;
  APPLATTR MeaQVersion,       BASEATTR version,         DATATYPE DT_STRING;
  APPLATTR MeaQDescription,   BASEATTR description,     DATATYPE DT_STRING;
  APPLATTR version_date,      BASEATTR version_date,    DATATYPE DT_DATE;
  APPLATTR Rank,              BASEATTR rank,            DATATYPE DT_LONG;
  APPLATTR Dimension,         BASEATTR dimension,       DATATYPE DT_LONG,
                                                      CARDINALITY 0,MANY;
  APPLATTR DataType,         BASEATTR datatype,       DATATYPE DT_STRING;
  APPLATTR TypeSize,         BASEATTR type_size,      DATATYPE DT_LONG;
  APPLATTR Interpolation,    BASEATTR interpolation,  DATATYPE DT_STRING;
  APPLATTR Minimum,          BASEATTR minimum,        DATATYPE DT_DOUBLE;
  APPLATTR Maximum,          BASEATTR maximum,        DATATYPE DT_DOUBLE;
  APPLATTR Average,          BASEATTR average,        DATATYPE DT_DOUBLE;
  APPLATTR SDeviation,   BASEATTR standard_deviation, DATATYPE DT_DOUBLE;
  APPLATTR MeaQuantStat,                              DATATYPE DT_SHORT;
  APPLATTR TimeOffset,                                DATATYPE DT_DOUBLE;
  APPLATTR SamplingRate,                              DATATYPE DT_DOUBLE;

  APPLATTR Measurement,      BASEATTR measurement,    REF_TO Measurement;
  APPLATTR LocalColumns,     BASEATTR local_columns,  REF_TO LocalColumn,
                                                      CARDINALITY 0,MANY;
  APPLATTR Quantity,         BASEATTR quantity,       REF_TO Quantity;
  APPLATTR Unit,             BASEATTR unit,           REF_TO Unit;
ENDAPPLELEM;


APPLELEM Submatrix,BASETYPE AoSubmatrix
  APPLATTR name,             BASEATTR name,           DATATYPE DT_STRING;
  APPLATTR id,               BASEATTR id,             DATATYPE DT_LONG;
  APPLATTR version,          BASEATTR version,        DATATYPE DT_STRING;
  APPLATTR description,      BASEATTR description,    DATATYPE DT_STRING;
  APPLATTR version_date,     BASEATTR version_date,   DATATYPE DT_DATE;
  APPLATTR number_of_rows,   BASEATTR number_of_rows, DATATYPE DT_LONG;
  APPLATTR local_columns,    BASEATTR local_columns,  REF_TO LocalColumn,
                                                      CARDINALITY 0,MANY;
  APPLATTR measurement,      BASEATTR measurement,    REF_TO Measurement;
ENDAPPLELEM;


APPLELEM LocalColumn,BASETYPE AoLocalColumn
  APPLATTR name,             BASEATTR name,           DATATYPE DT_STRING;
  APPLATTR id,               BASEATTR id,             DATATYPE DT_LONG;
  APPLATTR version,          BASEATTR version,        DATATYPE DT_STRING;
  APPLATTR description,      BASEATTR description,    DATATYPE DT_STRING;
  APPLATTR version_date,     BASEATTR version_date,   DATATYPE DT_DATE;
  APPLATTR flags,            BASEATTR flags,          DATATYPE DT_SHORT,
                                                      CARDINALITY 0,MANY;
  APPLATTR global_flag,      BASEATTR global_flag,    DATATYPE DT_SHORT;
  APPLATTR independent,      BASEATTR independent,    DATATYPE DT_BOOLEAN;
  APPLATTR minimum,          BASEATTR minimum,        DATATYPE DT_DOUBLE;
  APPLATTR maximum,          BASEATTR maximum,        DATATYPE DT_DOUBLE;
  APPLATTR values,           BASEATTR values,         DATATYPE DT_UNKNOWN,
                                                   CARDINALITY 0,MANY;
  APPLATTR submatrix,        BASEATTR submatrix,      REF_TO Submatrix;
  APPLATTR measurement_quantity,      BASEATTR measurement_quantity,
                                      REF_TO   MeaQuantity;
ENDAPPLELEM;

/*
//---------------------------------------------------------------------------
//
// Construction of a catalogue of quantities
// Application elements
//               Quantity   (-Group),Unit  (-Group), PhysDim,
//  of base type AoQuantity (-Group),AoUnit(-Group), AoPhysicalDimension
//
//---------------------------------------------------------------------------
*/

APPLELEM Quantity, BASETYPE AoQuantity
  APPLATTR QName,             BASEATTR name,           DATATYPE DT_STRING;
  APPLATTR QId,               BASEATTR id,             DATATYPE DT_LONGLONG;
  APPLATTR QVersion,          BASEATTR version,        DATATYPE DT_STRING;
  APPLATTR QDescription,      BASEATTR description,    DATATYPE DT_STRING;
  APPLATTR version_date,      BASEATTR version_date,   DATATYPE DT_DATE;
  APPLATTR DefaultRank,       BASEATTR default_rank,   DATATYPE DT_LONG;
  APPLATTR DefaultDimension,  BASEATTR default_dimension, DATATYPE DT_LONG,
                                                         CARDINALITY 0,MANY;
  APPLATTR DefaultDatatype, BASEATTR default_datatype,  DATATYPE DT_STRING;
  APPLATTR DefaultType_size,BASEATTR default_type_size, DATATYPE DT_LONG;
  APPLATTR DefaultMqName,   BASEATTR default_mq_name,   DATATYPE DT_STRING;

  APPLATTR DefaultUnit,     BASEATTR default_unit,    REF_TO Unit;
  APPLATTR Successors,      BASEATTR successors,      REF_TO Quantity,
                                                      CARDINALITY 0,MANY;
  APPLATTR Predecessor,     BASEATTR predecessor,     REF_TO Quantity;
  APPLATTR QuantityGroup,   BASEATTR groups,          REF_TO QuantityGroup,
                                                      CARDINALITY 0,MANY;
ENDAPPLELEM;


APPLELEM QuantityGroup, BASETYPE AoQuantityGroup
  APPLATTR QGroupName,        BASEATTR name,           DATATYPE DT_STRING;
  APPLATTR QGroupId,          BASEATTR id,             DATATYPE DT_LONGLONG;
  APPLATTR QGroupVersion,     BASEATTR version,        DATATYPE DT_STRING;
  APPLATTR QGroupDescription, BASEATTR description,    DATATYPE DT_STRING;
  APPLATTR version_date,      BASEATTR version_date,   DATATYPE DT_DATE;
  APPLATTR QantGroupState,                             DATATYPE DT_SHORT;
  APPLATTR ListOfQuantities,  BASEATTR quantities,    REF_TO Quantity,
                                                      CARDINALITY 0,MANY;
ENDAPPLELEM;


APPLELEM Unit, BASETYPE AoUnit
  APPLATTR UnitName,          BASEATTR name,          DATATYPE DT_STRING;
  APPLATTR UnitId,            BASEATTR id,            DATATYPE DT_LONGLONG;
  APPLATTR UnitVersion,       BASEATTR version,       DATATYPE DT_STRING;
  APPLATTR UnitDescription,   BASEATTR description,   DATATYPE DT_STRING;
  APPLATTR version_date,      BASEATTR version_date,  DATATYPE DT_DATE;
  APPLATTR UnitFactor,        BASEATTR factor,        DATATYPE DT_FLOAT;
  APPLATTR UnitOffset,        BASEATTR offset,        DATATYPE DT_FLOAT;

  APPLATTR PhysDim,           BASEATTR phys_dimension,REF_TO PhysDim;
  APPLATTR UnitGroups,        BASEATTR groups,        REF_TO UnitGroup,
                                                      CARDINALITY 0,MANY;
ENDAPPLELEM;


APPLELEM UnitGroup, BASETYPE AoUnitGroup
  APPLATTR UGroupName,        BASEATTR name,          DATATYPE DT_STRING;
  APPLATTR UGroupId,          BASEATTR id,            DATATYPE DT_LONGLONG;
  APPLATTR UGroupVersion,     BASEATTR version,       DATATYPE DT_STRING;
  APPLATTR UGroupDescription, BASEATTR description,   DATATYPE DT_STRING;
  APPLATTR version_date,      BASEATTR version_date,  DATATYPE DT_DATE;
  APPLATTR UnitGroupstate,                            DATATYPE DT_SHORT;

  APPLATTR Units,             BASEATTR units,         REF_TO Unit,
                                                      CARDINALITY 0,MANY;
ENDAPPLELEM;


APPLELEM PhysDim, BASETYPE AoPhysicalDimension
  APPLATTR PhysDimName,       BASEATTR name,          DATATYPE DT_STRING;
  APPLATTR PhysDimId,         BASEATTR id,            DATATYPE DT_LONGLONG;
  APPLATTR PhysDimVersion,    BASEATTR version,       DATATYPE DT_STRING;
  APPLATTR PhysDimDescription,BASEATTR description,   DATATYPE DT_STRING;
  APPLATTR version_date,      BASEATTR version_date,  DATATYPE DT_DATE;

  APPLATTR length,          BASEATTR length_exp,        DATATYPE DT_SHORT;
  APPLATTR mass,            BASEATTR mass_exp,          DATATYPE DT_SHORT;
  APPLATTR time,            BASEATTR time_exp,          DATATYPE DT_SHORT;
  APPLATTR current,         BASEATTR current_exp,       DATATYPE DT_SHORT;
  APPLATTR temperature,     BASEATTR temperature_exp,   DATATYPE DT_SHORT;
  APPLATTR molar,           BASEATTR molar_amount_exp,  DATATYPE DT_SHORT;
  APPLATTR luminous,        BASEATTR luminous_intensity_exp,
                                                        DATATYPE DT_SHORT;

  APPLATTR Units,           BASEATTR units,             REF_TO Unit,
                                                        CARDINALITY 0,MANY;
ENDAPPLELEM;

/*
//---------------------------------------------------------------------------
//
// Application elements User, UserGroup, ACLA, ACLI used for
//    security information
//
//---------------------------------------------------------------------------
*/

APPLELEM User, BASETYPE AoUser
  APPLATTR UserName,          BASEATTR name,          DATATYPE DT_STRING;
  APPLATTR UserId,            BASEATTR id,            DATATYPE DT_LONGLONG;
  APPLATTR UserVersion,       BASEATTR version,       DATATYPE DT_STRING;
  APPLATTR UserDescription,   BASEATTR description,   DATATYPE DT_STRING;
  APPLATTR version_date,      BASEATTR version_date,  DATATYPE DT_DATE;
  APPLATTR password,          BASEATTR password,      DATATYPE DT_STRING;
  APPLATTR groups,            BASEATTR groups,        REF_TO UserGroup,
                                                      CARDINALITY 0,MANY;
  APPLATTR UserDepartment,                            DATATYPE DT_STRING;
  APPLATTR AliasName,                                 DATATYPE DT_STRING;
ENDAPPLELEM;

APPLELEM UserGroup, BASETYPE AoUserGroup
  APPLATTR GroupName,         BASEATTR name,          DATATYPE DT_STRING;
  APPLATTR GroupId,           BASEATTR id,            DATATYPE DT_LONGLONG;
  APPLATTR GroupVersion,      BASEATTR version,       DATATYPE DT_STRING;
  APPLATTR GroupDescription,  BASEATTR description,   DATATYPE DT_STRING;
  APPLATTR version_date,      BASEATTR version_date,  DATATYPE DT_DATE;
  APPLATTR superuser_flag,    BASEATTR superuser_flag,DATATYPE DT_BOOLEAN;
  APPLATTR users,             BASEATTR users,         REF_TO User,
                                                      CARDINALITY 0,MANY;
ENDAPPLELEM;

/*
//---------------------------------------------------------------------------
//
//  ACLA, ACLI, ACLTemplate are NOT application elements, instead belong
//  only to the base element, therefore NOT included here.
```

```
//
//  SecurityLevel, InitialRightsAt are NOT application elements, instead
//  belong only to the base element, therefore NOT included here.
//
//---------------------------------------------------------------------------
*/

/*
//***************************************************************************
//
// Instance elements section
//
//***************************************************************************
*/
/*
//---------------------------------------------------------------------------
// Instance elements of the application element PhysDim
//                               from base type AoPhysicalDimension
//---------------------------------------------------------------------------
*/

INSTELEM PhysDim
  PhysDimName        = "empty ";
  PhysDimId          = 1;
  PhysDimVersion     = "1.0";
  PhysDimDescription = "No dimension";
  version_date       = "199702010900";

  length             = 0;   // Meter      m
  mass               = 0;   // Kilogramme kg
  time               = 0;   // Second     s
  current            = 0;   // Ampere     A
  temperature        = 0;   // Kelvin     K
  molar              = 0;   // mol        mol
  luminous           = 0;   // Candela    cd

  units              = 1;   /* Refs to instance elements of Unit */
ENDINSTELEM;


INSTELEM PhysDim
  PhysDimName        = "temperature";
  PhysDimId          = 2;
  PhysDimVersion     = "1.0";
  PhysDimDescription = "temperature dimension";
  version_date       = "199702010900";
  length             = 0;   // Meter      m
  mass               = 0;   // Kilogramme kg
  time               = 0;   // Second     s
  current            = 0;   // Ampere     A
  temperature        = 1;   // Kelvin     K
  molar              = 0;   // mol        mol
  luminous           = 0;   // Candela    cd

  units              = 2,3; /* Refs to instance elements of Unit */
ENDINSTELEM;


INSTELEM PhysDim
  PhysDimName        = "frequency";
  PhysDimId          = 3;
  PhysDimVersion     = "1.0";
  PhysDimDescription = "frequency dimension";
  version_date       = "199702010900";

  length             = 0;   // Meter      m
  mass               = 0;   // Kilogramme kg
  time               = -1;  // Second     s
  current            = 0;   // Ampere     A
  temperature        = 0;   // Kelvin     K
  molar              = 0;   // mol        mol
  luminous           = 0;   // Candela    cd

  units              = 4,5; /* Refs to instance elements of Unit */
ENDINSTELEM;


INSTELEM PhysDim
  PhysDimName        = "work";
  PhysDimId          = 4;
  PhysDimVersion     = "1.0";
  PhysDimDescription = "work dimension";
  version_date       = "199702010900";

  length             = 2;   // Meter      m
  mass               = 1;   // Kilogramme kg
  time               = -2;  // Second     s
  current            = 0;   // Ampere     A
  temperature        = 0;   // Kelvin     K
  molar              = 0;   // mol        mol
  luminous           = 0;   // Candela    cd
  units              = 6;   /* Refs to instance elements of Unit */
ENDINSTELEM;


INSTELEM PhysDim
  PhysDimName        = "pressure";
  PhysDimId          = 5;
  PhysDimVersion     = "1.0";
  PhysDimDescription = "pressure dimension";
  version_date       = "199702010900";

  length             = -1;  // Meter      m
  mass               = 1;   // Kilogramme kg
  time               = -2;  // Second     s
  current            = 0;   // Ampere     A
  temperature        = 0;   // Kelvin     K
  molar              = 0;   // mol        mol
  luminous           = 0;   // Candela    cd

  units              = 7,8; /* Refs to instance elements of Unit */
ENDINSTELEM;

/*
//---------------------------------------------------------------------------
// Instance elements of the application element Unit
//                               from base type AoUnit
//---------------------------------------------------------------------------
*/

INSTELEM Unit
  UnitName        = "-";
  UnitId          = 1;
  UnitVersion     = "1.0";
  UnitDescription = "Empty Unit";
  version_date    = "199702010900";
  UnitFactor      = 1.;
  UnitOffset      = 0.;

  PhysDim         = 1;         /* Ref  to instance element  of PhysDim   */
  UnitGroups      = UNDEFINED; /* Refs to instance elements of UnitGroup */
ENDINSTELEM;


INSTELEM Unit
  UnitName        = "Grd C";
  UnitId          = 2;
  UnitVersion     = "1.0";
  UnitDescription = "Grad Celcius";
  version_date    = "199702010900";
  UnitFactor      = 1.0;
  UnitOffset      = 273.15;

  PhysDim         = 2;         /* Ref  to instance element  of PhysDim   */
  UnitGroups      = UNDEFINED; /* Refs to instance elements of UnitGroup */
ENDINSTELEM;


INSTELEM Unit
  UnitName        = "Grd K";
  UnitId          = 3;
  UnitVersion     = "1.0";
  UnitDescription = "Grad Kelvin";
  version_date    = "199702010900";
  UnitFactor      = 1.0;
  UnitOffset      = 0.;

  PhysDim         = 2;         /* Ref  to instance element  of PhysDim   */
  UnitGroups      = UNDEFINED; /* Refs to instance elements of UnitGroup */
ENDINSTELEM;


INSTELEM Unit
  UnitName        = "1/min";
  UnitId          = 4;
  UnitVersion     = "1.0";
  UnitDescription = "Revolution per minunte";
  version_date    = "199702010900";
  UnitFactor      = 0.016667;
  UnitOffset      = 0.;

  PhysDim         = 3;         /* Ref  to instance element  of PhysDim   */
  UnitGroups      = UNDEFINED; /* Refs to instance elements of UnitGroup */
ENDINSTELEM;


INSTELEM Unit
  UnitName        = "1/sec";
  UnitId          = 5;
  UnitVersion     = "1.0";
  UnitDescription = "Revolution per second";
  version_date    = "199702010900";
  UnitFactor      = 1.0;
  UnitOffset      = 0.;

  PhysDim         = 3;         /* Ref  to instance element  of PhysDim   */
  UnitGroups      = UNDEFINED; /* Refs to instance elements of UnitGroup */
ENDINSTELEM;


INSTELEM Unit
  UnitName        = "Nm";
  UnitId          = 6;
  UnitVersion     = "1.0";
  UnitDescription = "Torque in Nm";
  version_date    = "199702010900";
  UnitFactor      = 1.;
  UnitOffset      = 0.;

  PhysDim         = 4;         /* Ref  to instance element  of PhysDim   */
  UnitGroups      = UNDEFINED; /* Refs to instance elements of UnitGroup */
ENDINSTELEM;


INSTELEM Unit
  UnitName        = "mbar";
  UnitId          = 7;
  UnitVersion     = "1.0";
  UnitDescription = "Pressure in mbar";
  version_date    = "199702010900";
  UnitFactor      = 100.;
  UnitOffset      = 0.;

  PhysDim         = 5;         /* Ref  to instance element  of PhysDim   */
  UnitGroups      = UNDEFINED; /* Refs to instance elements of UnitGroup */
ENDINSTELEM;


INSTELEM Unit
  UnitName        = "bar";
  UnitId          = 8;
  UnitVersion     = "1.0";
  UnitDescription = "Pressure in bar";
  version_date    = "199702010900";
  UnitFactor      = 100000.;
  UnitOffset      = 0.;

  PhysDim         = 5;         /* Ref  to instance element  of PhysDim   */
  UnitGroups      = UNDEFINED; /* Refs to instance elements of UnitGroup */
ENDINSTELEM;

/*
//---------------------------------------------------------------------------
// Instance elements of the application element Quantity
//                               from base type AoQuantity
//---------------------------------------------------------------------------
*/

INSTELEM Quantity
  QName            = "TEMP";
  QId              = 1;
  QVersion         = "1.1";
  QDescription     = "Temperature Quantity central";
  version_date     = "199703100900";
  DefaultRank      = 0;
  DefaultDimension = UNDEFINED;
  DefaultDatatype  = "DT_FLOAT";
  DefaultType_size = UNDEFINED;
  DefaultMqName    = "TEMP";

  DefaultUnit      = 2;         /* Ref  to instance element  of Unit     */
  Predecessor      = UNDEFINED; /* Ref  to instance element  of Quantity */
  Successors       = 2,3;       /* Refs to instance elements of Quantity */
  QuantityGroup    = UNDEFINED; /* Ref  to instance els.of QuantityGroup */
ENDINSTELEM;


INSTELEM Quantity
  QName            = "LEFT";
  QId              = 2;
  QVersion         = "1.1";
  QDescription     = "Temperature Quantity left";
  version_date     = "199703100900";
  DefaultRank      = 0;
  DefaultDimension = UNDEFINED;
  DefaultDatatype  = "DT_FLOAT";
  DefaultType_size = UNDEFINED;
  DefaultMqName    = "TEMP_LEFT";

  DefaultUnit      = 2;         /* Ref  to instance element  of Unit     */
  Predecessor      = 1;         /* Ref  to instance element  of Quantity */
  Successors       = UNDEFINED; /* Refs to instance elements of Quantity */
  QuantityGroup    = UNDEFINED; /* Ref  to instance els.of QuantityGroup */
ENDINSTELEM;
```

```
INSTELEM Quantity
  QName            = "RIGHT";
  QId              = 3;
  QVersion         = "1.1";
  QDescription     = "Temperature Quantity right";
  version_date     = "199703100900";
  DefaultRank      = 0;
  DefaultDimension = UNDEFINED;
  DefaultDatatype  = "DT_FLOAT";
  DefaultType_size = UNDEFINED;
  DefaultMqName    = "TEMP_RIGHT";

  DefaultUnit      = 2;         /* Ref  to instance element  of Unit     */
  Predecessor      = 1;         /* Ref  to instance element  of Quantity */
  Successors       = UNDEFINED; /* Refs to instance elements of Quantity */
  QuantityGroup    = UNDEFINED; /* Ref  to instance els.of QuantityGroup */
ENDINSTELEM;


INSTELEM Quantity
  QName            = "SPTNR";
  QId              = 4;
  QVersion         = "1.1";
  QDescription     = "System point number";
  version_date     = "199703100900";
  DefaultRank      = 0;
  DefaultDimension = UNDEFINED;
  DefaultDatatype  = "DT_LONG";
  DefaultType_size = UNDEFINED;
  DefaultMqName    = "SPTNR";

  DefaultUnit      = 1;         /* Ref  to instance element  of Unit     */
  Predecessor      = UNDEFINED; /* Ref  to instance element  of Quantity */
  Successors       = UNDEFINED; /* Refs to instance elements of Quantity */
  QuantityGroup    = UNDEFINED; /* Ref  to instance els.of QuantityGroup */
ENDINSTELEM;


INSTELEM Quantity
  QName            = "N";
  QId              = 5;
  QVersion         = "1.1";
  QDescription     = "Revolution";
  version_date     = "199703100900";
  DefaultRank      = 0;
  DefaultDimension = UNDEFINED;
  DefaultDatatype  = "DT_FLOAT";
  DefaultType_size = UNDEFINED;
  DefaultMqName    = "N";

  DefaultUnit      = 4;         /* Ref  to instance element  of Unit     */
  Predecessor      = UNDEFINED; /* Ref  to instance element  of Quantity */
  Successors       = UNDEFINED; /* Refs to instance elements of Quantity */
  QuantityGroup    = UNDEFINED; /* Ref  to instance els.of QuantityGroup */
ENDINSTELEM;


INSTELEM Quantity
  QName            = "MD";
  QId              = 6;
  QVersion         = "1.1";
  QDescription     = "Torque";
  version_date     = "199703100900";
  DefaultRank      = 0;
  DefaultDimension = UNDEFINED;
  DefaultDatatype  = "DT_FLOAT";
  DefaultType_size = UNDEFINED;
  DefaultMqName    = "MD";

  DefaultUnit      = 6;         /* Ref  to instance element  of Unit     */
  Predecessor      = UNDEFINED; /* Ref  to instance element  of Quantity */
  Successors       = UNDEFINED; /* Refs to instance elements of Quantity */
  QuantityGroup    = UNDEFINED; /* Ref  to instance els.of QuantityGroup */
ENDINSTELEM;


INSTELEM Quantity
  QName            = "PL";
  QId              = 7;
  QVersion         = "1.1";
  QDescription     = "Pressure left";
  version_date     = "199703100900";
  DefaultRank      = 0;
  DefaultDimension = UNDEFINED;
  DefaultDatatype  = "DT_FLOAT";
  DefaultType_size = UNDEFINED;
  DefaultMqName    = "PL";

  DefaultUnit      = 7;         /* Ref  to instance element  of Unit     */
  Predecessor      = UNDEFINED; /* Ref  to instance element  of Quantity */
  Successors       = UNDEFINED; /* Refs to instance elements of Quantity */
  QuantityGroup    = UNDEFINED; /* Ref  to instance els.of QuantityGroup */
ENDINSTELEM;

/*
//---------------------------------------------------------------------------
//
// Instance elements of the application elements Engine and Test
//                                from base type AoTest and AoSubTest
//
//---------------------------------------------------------------------------
*/

INSTELEM Engine
  EngineName        = "TestEngine 1";
  EngineId          = 1;
  EngineVersion     = "A 1.0";
  EngineDescription = "The first test engine";
  CreateDate        = "19971209141345";
  EngineType         = "ABC4711";  /* Hel */
  bore              = 92.;
  cylindernumber    = 6;
  stroke            = 95.3;

  SubTests          = 1;    // Ref  to instance element  of Test
  EngineUser        = 21;   // Ref  to instance element  of User
ENDINSTELEM;


INSTELEM Test
  TestName          = "Test-Configuration 1";
  TestId            = 1;
  TestVersion       = "B20";
  TestDescription   = "The first test of engine 1";
  CreateDate        = "19971210101422";
  TestType          = "Functionality Test";
  TestComment       = "Test: torque and at special points temperature";
  exhaust           = "System 4711";
  air_filter        = "C 12.1";

  Measurements      = 1,2; /* Refs to instance elements of Measurement */
  MainTest          = 1;   /* Ref  to instance element  of Engine      */
  TestUser /* Hel */= 22;  /* Ref  to instance element  of User        */
ENDINSTELEM;

/*
```

```
//-----------------------------------------------------------------------------
//
// Instance elements of the application structure part with
//    Measurement, MeaQuantity, Submatrix, LocalColumn with value_sequence
// for the first measurement
//
//-----------------------------------------------------------------------------
*/

INSTELEM Measurement
  MeaName            = "M001A";
  MeaId              = 1;
  MeaVersion         = "A1";
  MeaDescription     = "1. Measurement ";
  MeasurementBegin   = "19971210141345";
  MeasurementEnd     = "19971210162135";
  CreateDate         = "19971210165331"; /* Hel */
  MeaType            = "TLM";
  MeaComment         = "No Problem";
  ROZ                = UNDEFINED;

  Test               = 1;    /* Ref  to instance element of Test        */
  MeaQuantities      = 1,2,3;/* Ref  to instance element of MeaQuantity */
  Submatrices        = 1;    /* Ref  to instance element of Submatrix   */
ENDINSTELEM;


INSTELEM MeaQuantity
  MeaQName          = "N";
  MeaQId            = 1;
  MeaQVersion       = "D1";
  MeaQDescription   = "Number of revolutions";
  Rank              = 0;
  Dimension         = UNDEFINED;
  DataType          = "DT_FLOAT";
  TypeSize          = UNDEFINED;
  Minimum           = 500.;
  Maximum           = 3200.;
  Average           = 1850.;

  Measurement       = 1;  /* Ref  to instance element  of Measurement */
  LocalColumns      = 1;  /* Refs to instance elements of LocalColumn */
  Quantity          = 5;  /* Ref  to instance element  of Quantity    */
  Unit              = 4;  /* Ref  to instance element  of Unit        */
ENDINSTELEM;
```

```
INSTELEM MeaQuantity
  MeaQName         = "MD";
  MeaQId           = 2;
  MeaQVersion      = "D";
  MeaQDescription  = "Torque";
  Rank             = 0;
  Dimension        = UNDEFINED;
  DataType         = "DT_FLOAT";
  Minimum          = 150.;
  Maximum          = 255.;
  Average          = 190.;
  SDeviation       = 22.2;

  Measurement      = 1;  /* Ref  to instance element  of Measurement */
  LocalColumns     = 2;  /* Refs to instance elements of LocalColumn */
  Quantity         = 6;  /* Ref  to instance element  of Quantity    */
  Unit             = 6;  /* Ref  to instance element  of Unit        */
ENDINSTELEM;


INSTELEM MeaQuantity
  MeaQName         = "PL";
  MeaQId           = 3;
  MeaQVersion      = "D";
  MeaQDescription  = "Pressure ";
  Rank             = 0;
  Dimension        = UNDEFINED;
  DataType         = "DT_FLOAT";
  Minimum          = 991.2;
  Maximum          = 993.2;
  Average          = 992.0;

  Measurement      = 1;  /* Ref  to instance element  of Measurement */
  LocalColumns     = 3;  /* Refs to instance elements of LocalColumn */
  Quantity         = 7;  /* Ref  to instance element  of Quantity    */
  Unit             = 7;  /* Ref  to instance element  of Unit        */
ENDINSTELEM;


INSTELEM Submatrix
  name           = "M001A";
  id             = 1;
  version        = "1a";
  description    = "1. Measurement for data transfer via ASAM";
  version_date   = UNDEFINED;
  measurement    = 1;
  number_of_rows = 15;
  local_columns  = 1,2,3; /* Refs to instance elements of LocalColumn */
ENDINSTELEM;


INSTELEM LocalColumn
  name        = "N";
  id          = 1;
  flags       = 0;
  global_flag = 0;
  independent = TRUE;
  minimum     = 500.;
  maximum     = 3200.;

  measurement_quantity = 1;  /* Ref to instance element of MeaQuantity */
  submatrix = 1;             /* Ref to instance element of Submatrix   */
  values = DATATYPE DT_FLOAT
          ,500.,700.,900.,1100.,1300.,1400.,1600.,1800.,2000.
          ,2200.,2400.,2600.,2800.,3000.,3200.;
ENDINSTELEM;


INSTELEM LocalColumn
  name        = "MD";
  id          = 2;
  flags       = 0;
  global_flag = 0;
  independent = FALSE;
  minimum     = 150.;
  maximum     = 255.;

  measurement_quantity = 2;  /* Ref to instance element of MeaQuantity */
  submatrix = 1;             /* Ref to instance element of Submatrix   */
  values = DATATYPE DT_FLOAT
          ,150.,160.,170.,180.,190.,200.,210.,220.
          ,230.,240.,250.,255.,250.,230.,190.;
ENDINSTELEM;


INSTELEM LocalColumn
  name        = "PL";
  id          = 3;
  flags       = 0;
  global_flag = 0;
  independent = FALSE;
  minimum     = 991.2;
  maximum     = 993.2;

  measurement_quantity = 3;  /* Ref to instance element of MeaQuantity  */
  submatrix = 1;             /* Ref to instance element of AoSubmatrix  */
  values = DATATYPE DT_FLOAT
          ,991.2,991.7,991.5,991.9,992.1,987.6,992.2,993.
          ,993.2,993.2,992.7,993.,993.,992.7,992.7;
ENDINSTELEM;

/*
//---------------------------------------------------------------------------
//
// Instance elements of the application structure part with
//    Measurement, MeaQuantity, Submatrix, LocalColumn with value_sequence
// for the second measurement
//
//---------------------------------------------------------------------------
*/

INSTELEM Measurement
  MeaName           = "M002A";
  MeaId             = 2;
  MeaVersion        = "A1";
  MeaDescription    = "2. Measurement ";
  MeasurementBegin  = "19971211090412";
  MeasurementEnd    = "19971211121145";
  CreateDate        = "19971211121522"; /* Hel */
  MeaType           = "VLM";
  MeaComment        = "No problem";
  ROZ               = UNDEFINED;

  Test              = 1;      /* Ref to instance element  of Test        */
  MeaQuantities     = 4,5,6,7;/* Ref to instance element  of MeaQuantity */
  Submatrices       = 2,3;    /* Ref to instance elements of Submatrix   */
ENDINSTELEM;


INSTELEM MeaQuantity
  MeaQName         = "SPTNR";
  MeaQId           = 4;
  MeaQVersion      = "A";
  MeaQDescription  = "System point number ";
  Rank             = 0;
  Dimension        = UNDEFINED;
  DataType         = "DT_LONG";
  Minimum          = 1.0;
  Maximum          = 15.;
  Average          = 7.5;

  Measurement      = 2;     /* Ref to instance element  of Measurement */
  LocalColumns     = 4,7;   /* Ref to instance elements of LocalColumn */
  Quantity         = 4;     /* Ref to instance element  of Quantity    */
  Unit             = 1;     /* Ref to instance element  of Unit        */
ENDINSTELEM;


INSTELEM MeaQuantity
  MeaQName         = "N";
  MeaQId           = 5;
  MeaQVersion      = "D";
  MeaQDescription  = "Revolution ";
  Rank             = 0;
  DataType         = "DT_FLOAT";
  Minimum          = 500.;
  Maximum          = 3200.;
  Average          = 1850.;

  Measurement      = 2;    /* Ref  to instance element  of Measurement   */
  LocalColumns     = 5;    /* Refs to instance elements of LocalColumn   */
  Quantity         = 5;    /* Ref  to instance element  of Quantity      */
  Unit             = 4;    /* Ref  to instance element  of Unit          */
ENDINSTELEM;


INSTELEM MeaQuantity
  MeaQName         = "MD";
  MeaQId           = 6;
  MeaQVersion      = "D";
  MeaQDescription  = "Torque ";
  Rank             = 0;
  Dimension        = UNDEFINED;
  DataType         = "DT_FLOAT";
  Minimum          = 152.;
  Maximum          = 257.;
  Average          = 194.;

  Measurement      = 2;    /* Ref  to instance element  of Measurement */
  LocalColumns     = 6;    /* Refs to instance elements of LocalColumn */
  Quantity         = 6;    /* Ref  to instance element  of Quantity    */
  Unit             = 6;    /* Ref  to instance element  of Unit        */
ENDINSTELEM;
```

```
INSTELEM MeaQuantity
  MeaQName         = "TEMP1";
  MeaQId           = 7;
  MeaQVersion      = "D";
  MeaQDescription  = "Temperature ";
  Rank             = 0;
  Dimension        = UNDEFINED;
  DataType         = "DT_FLOAT";
  Minimum          = 150.;
  Maximum          = 183.;
  Average          = 169.;

  Measurement      = 2;    /* Ref  to instance element  of Measurement */
  LocalColumns     = 8;    /* Refs to instance elements of LocalColumn */
  Quantity         = 2;    /* Ref  to instance element  of Quantity    */
  Unit             = 2;    /* Ref  to instance element  of Unit        */
ENDINSTELEM;


INSTELEM Submatrix
  name           = "M002A";
  id             = 2;
  version        = "1a";
  description    = "2. Measurement for data transfer via ASAM";
  version_date   = UNDEFINED;
  measurement    = 2;
  number_of_rows = 15;
  local_columns  = 4,5,6;  /* Refs to instance elements of LocalColumn */
ENDINSTELEM;


INSTELEM LocalColumn
  name        = "SPKTNR";
  id          = 4;
  flags       = 0;
  global_flag = 0;
  independent = TRUE;
  minimum     = 1;
  maximum     = 15;

  measurement_quantity = 4;   /* Ref to instance element of MeaQuantity */
  submatrix = 2;              /* Ref to instance element of Submatrix   */
  values = DATATYPE DT_SHORT
          ,1, 2, 3, 4, 5, 6, 7, 8
          ,9,10,11,12,13,14,15;
ENDINSTELEM;


INSTELEM LocalColumn
  name        = "N";
  id          = 5;
  flags       = 0;
  global_flag = 0;
  independent = FALSE;
  minimum     = 500.;
  maximum     = 3200.;

  measurement_quantity = 5;  /* Ref to instance element of MeaQuantity */
  submatrix = 2;             /* Ref to instance element of Submatrix   */
  values = DATATYPE DT_FLOAT
          ,500.,700.,900.,1100.,1300.,1400.,1600.,1800.
          ,2000.,2200.,2400.,2600.,2800.,3000.,3200.;
ENDINSTELEM;


INSTELEM LocalColumn
  name        = "MD";
  id          = 6;
  flags       = 0;
  global_flag = 0;
  independent = FALSE;
  minimum     = 150.;
  maximum     = 255.;

  measurement_quantity = 6;  /* Ref to instance element of MeaQuantity */
  submatrix = 2;             /* Ref to instance element of Submatrix   */
  values = DATATYPE DT_FLOAT
          ,150.,160.,170.,180.,190.,200.,210.,220.
          ,230.,240.,250.,255.,250.,230.,190.;
ENDINSTELEM;


INSTELEM Submatrix
  name           = "M002B";
  id             = 3;
  version        = "1a";
  description    = "2. Measurement for data transfer via ASAM";
  version_date   = UNDEFINED;
  measurement    = 2;
  number_of_rows = 3;
  local_columns  = 7,8;   /* Refs instance elements of LocalColumn */
ENDINSTELEM;


INSTELEM LocalColumn
  name        = "SPKTNR";
  id          = 7;
  flags       = 0;
  global_flag = 0;
  independent = TRUE;
  minimum     = 5;
  maximum     = 15;

  measurement_quantity = 4;  /* Ref to instance element of MeaQuantity */
  submatrix = 2;             /* Ref to instance element of Submatrix   */
  values = DATATYPE DT_SHORT
          ,5,10,15;
ENDINSTELEM;


INSTELEM LocalColumn
  name        = "TEMP1";
  id          = 8;
  flags       = 0;
  global_flag = 0;
  independent = FALSE;
  minimum     = 150.;
  maximum     = 183.;

  measurement_quantity = 7;  /* Ref to instance element of MeaQuantity */
  submatrix = 2;             /* Ref to instance element of Submatrix   */
  values = DATATYPE DT_FLOAT
          ,150.,174.,183.;
ENDINSTELEM;

/*
//-------------------------------------------------------
//
// instance elements of the application elements User and UserGroup
// and ACLs for security control of application elements and instances.
//
//-------------------------------------------------------
*/

INSTELEM User
  UserName        = "Peter Sellers";
  UserId          = 21;
  UserVersion     = "1A";
  UserDescription = "Super user group 1";
  version_date    = "199708101105";
  UserDepartment  = "OP/EFG";
  AliasName       = "PS";
  password        = "^1kjws7hxoish^mjkkdjxoöe--@@@kjg798xh0880djdj90";
  groups          = 30,50;
ENDINSTELEM;

INSTELEM User
  UserName        = "Todd Martin";
  UserId          = 22;
  UserVersion     = "1B";
  UserDescription = "Sub user group 1b";
  version_date    = "199708101105";
  UserDepartment  = "OP/ABC";
  AliasName       = "TM";
  password        = "8snn236ikl=(/($mm22&/&)()NBT/(nger%/0!cv3kjkkkj";
  groups          = 30,40;
ENDINSTELEM;

INSTELEM User
  UserName        = "Otto Biermann";
  UserId          = 23;
  UserVersion     = "1A";
  UserDescription = "Super user group 1";
  version_date    = "199708101105";
  UserDepartment  = "OP/EFG";
  AliasName       = "OB";
  password        = "89xzhuon3m,ö))?imi/U(/%unpo39048ejdkßdksosüüoü3";
  groups          = 40,50;
ENDINSTELEM;

INSTELEM UserGroup
  GroupName       = "AQ 1";
  GroupId         = 30;
  GroupVersion    = "00";
  GroupDescription= "Group 1";
  version_date    = "199708101105";
  superuser_flag  = FALSE;
  users           = 21,22;
ENDINSTELEM;

INSTELEM UserGroup
  GroupName       = "AQ 2";
  GroupId         = 40;
  GroupVersion    = "00";
  GroupDescription= "Group 2";
  version_date    = "199708101105";
  superuser_flag  = FALSE;
  users           = 22,23;
ENDINSTELEM;

INSTELEM UserGroup
  GroupName       = "TZU 23";
  GroupId         = 50;
  GroupVersion    = "00";
  GroupDescription= "Group 3";
  version_date    = "199708101105";
  superuser_flag  = TRUE;
  users           = 21,23;
ENDINSTELEM;

INSTELEM ACLA
  users             = REF_TYPE UserGroup 30;
  appl_element_name = "Engine";
  rights            = 7;
  attribute_name    = UNDEFINED;  // protection of whole application element
ENDINSTELEM;

INSTELEM ACLA
  users             = REF_TYPE UserGroup 30;
  appl_element_name = "Quantity";
  rights            = 3;
  attribute_name    = "description";  // protection of a specific attribute
ENDINSTELEM;

INSTELEM ACLA
  users             = REF_TYPE UserGroup 40;
  appl_element_name = "Measurement";
  rights            = 10;
  attribute_name    = UNDEFINED;  // protection of whole application element
ENDINSTELEM;

INSTELEM ACLI
  users             = REF_TYPE UserGroup 50;
  appl_element_name = "MeaQuantity";
  rights            = 1;
  instance_id       = 3;   // protection of an instance element
ENDINSTELEM;
```

```
INSTELEM ACLI
  users             = REF_TYPE UserGroup 40;
  appl_element_name = "MeaQuantity";
  rights            = 3;
  instance_id       = 5;   // protection of an instance element
ENDINSTELEM;

/*
//-------------------------------------------------------
// Initial rights (ACL templates) and their location
// and security level
//-------------------------------------------------------
*/

INSTELEM ACLTemplate
  users             = REF_TYPE UserGroup 30;
  appl_element_name = "Measurement";
  rights            = 3;
  instance_id       = 2;
  ref_appl_elem_name= "Test";
ENDINSTELEM;

INSTELEM ACLTemplate
  users             = REF_TYPE UserGroup 40;
  appl_element_name = "Measurement";
  rights            = 1;
  instance_id       = 2;
  ref_appl_elem_name= "Test";
ENDINSTELEM;

INSTELEM InitialRightsAt
  appl_element_name = "Measurement";
  attribute_name    = "Test";
ENDINSTELEM;

INSTELEM SecurityLevel
  appl_element_name = "Measurement";
  level             = 7; /* all or specific instances and even attributes */
ENDINSTELEM;

INSTELEM SecurityLevel
  appl_element_name = "Engine";
  level             = 1; /* only all instances */
ENDINSTELEM;

ATF_END;
```

5.11 BIBLIOGRAPHY

[API-DEF]	Manfred Keul, Stefan Sträßer Die Applikationsschnittstelle: Definition, Rev 1.1 TecMath, 1994
[API-KON]	Dr. Frank Bomarius, Manfred Keul, Stefan Sträßer Die Applikationsschnittstelle: Bedarfsanalyse und Design, Rev 1.4 TecMath, 1994
[ASAM-AOP]	Hermann Schäffler Protokollschicht und physikalische Ablage, Rev 1.2 Digital-PCS, 1995
[ASAM ODS]	Arbeitskreis zur Standardisierung von Automatisierungs- und Meßsystemen, Offline-Daten-Schnittstelle ASAM-AK, 1994
[ATF-GFS]	Martin Winkler Die ATF-Datei (ASAM Transport Format) GfS, 1995
[IEEE-754]	IEEE Standard for Binary Floating-Point Arithmetic ANSI/IEEE Std. 754-1985 IEEE; New York, 1985
[ODS-FLAT]	Horst Fiedler Internals FlatFile (Server light) AVL, 1996
[PROT]	Horst Fiedler Protocol Layer Interface and Physical Storage Definitions ASAM ODS, 1996
[RSA-IMPL]	P. Dornhofer Puma 4.7 Result Storage Architecture (RSA): Implementation AVL/PECD, 1995
[UNICODE]	The Unicode Standard: Worldwide Character Encoding, Version 1.0 Volume 1: ISBN 0-201-56788-1 Volume 2: ISBN 0-201-60845-6 1995
[WWW-HTML]	Russ Jones, Adrian Nye, Übersetzer: Thomas Merz HTML und das World Wide Web ISBN3-930673-34-7 O'Reilly/International Thomson Verlag 1995

5.12 REVISION HISTORY

Date Editor	Changes
2003-06 P. Voltmann	Handling of security information added
2003-10-07 R. Bartz	Some errors have been fixed “ATF 1.4.1” has been renamed to “ATF/CLA” (classic)
2003-12 R. Bartz	List of “known issues” has been extended Explanations of BYTESTR and BLOB have been adapted Minor textual changes have been introduced
2003-12-30 R. Bartz	The **Release** version has been created
2004-05 R. Bartz	Minor textual changes have been introduced

5.12.1 PREVIOUS CHANGES

- Keywords for Infinity and NaN
- Correction of syntax diagrams for integer and real
- Enable complex values
- Allow ASAM Path instead of id
- New examples
- Renamed (D)COMPLEX, syntax diagram added
- Bytestream added, including syntax diagram
- .CR. in strings
- Value sequences in Local Columns
- Modified use of umlauts, extended class of usable characters
- More precise description of components, in particular binary components (Dec. 1999)
- Data type DT_UNKNOWN for LocalColumn added (Jan. 2000)
- Data type DT_EXTERNALREFERENCE added (Jan. 2000)
- Enumeration extended (June 2000)
- Security added (June 2000)
- Security updated (September 2000)

5.12.2 KNOWN ISSUES

The following issues are known. As ASAM ODS decided to not further change the technical specification of the ATF/CLA, they are not addressed in the specification. However anyone implementing ATF/CLA should make sure that his implementation is not affected by these issues.

- Potential rounding errors with float and double values may be eliminated by using IEEE conform bit structures in hex representation.
- The following keywords are not contained in the list of reserved keywords in section 5.5 but should also not be used as identifiers for application or instance elements: DT_ENUM, DT_EXTERNALREFERENCE, DT_LONG_BEO, DT_LONGLONG_BEO, DT_SHORT_BEO, IEEEFLOAT4_BEO, IEEEFLOAT8_BEO
- The following keywords are (erroneously) contained in the list of reserved keywords in section 5.5; they may be used as identifiers for application or instance elements: DT_CFLOAT, DT_CDOUBLE

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 6
ATF/XML

Version 1.0

Association for **S**tandardization of
Automation and **M**easuring Systems

Dated: 30.09.2004

Status of Document

Reference:	ASAM ODS Version 5.0 ATF/XML
Date:	30.09.2004
Author:	Mark Quinsland, HighQSoft; Dr. Helmut Helpenstein, National Instruments
Type:	Specification
Doc-ID:	ASAM_ODS_CH06_ATF_XML.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 (odsftr@asam.net) to make sure this issue will be addressed within the next review cycle.

Contents

6 THE ASAM TRANSPORT FORMAT IN XML (ATF/XML) 6-3

6.1 INTRODUCTION 6-3
6.2 ATF/XML AND ATF/CLA COMPARISON 6-3
6.3 STRUCTURE OF AN ATF/XML FILE 6-6
6.4 LOGICAL AND PHYSICAL ATF/XML FILES 6-6
6.4.1 USING XINCLUDE TO COMBINE MULTIPLE XML DOCUMENTS 6-7
6.4.2 DECLARING THE ROOT ELEMENT 6-8
6.4.3 DECLARING THE LOCALE 6-9
6.4.4 DECLARING THE ODS BASE MODEL 6-9
6.4.5 USING EXTERNAL FILES FOR MEASUREMENT DATA 6-9
6.5 VALIDATING AN ATF/XML FILE 6-12
6.5.2 CREATING AN APPLICATION SCHEMA 6-14
6.6 DEFINING AN APPLICATION MODEL 6-16
6.7 USING SPACES IN ELEMENT & ATTRIBUTE NAMES 6-22
6.8 DECLARING INSTANCE DATA 6-24
6.8.2 DECLARING INSTANCE ATTRIBUTES 6-26
6.8.3 USING EXTERNAL FILE COMPONENTS FOR MEASUREMENT DATA 6-27
6.9 SECURITY 6-38
6.9.1 STORING SECURITY INFORMATION 6-38
6.10 DATA TYPE USAGE IN ATF/XML 6-41
6.11 ODS BASE SCHEMA FILE 6-43
6.12 EXAMPLE ATF/XML FILE 6-66
6.13 KNOWN PROBLEMS 6-94
6.14 ADDITIONAL RESOURCES 6-94
6.15 REVISION HISTORY 6-94

Scope

This document describes the ASAM Transport Format in XML (ATF/XML) of the ASAM ODS Version 5.0.

Intended Audience

This document is intended for implementers of ASAM ODS Version 5.0. It shall be used as a technical reference with examples how to write ATF files in XML with the required information used in 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

6 THE ASAM TRANSPORT FORMAT IN XML (ATF/XML)

6.1 INTRODUCTION

The ASAM ODS specification calls for a file storage system that allows measurement data to be stored independent of a database. Prior to version 5.0 of ASAM ODS, that file storage system was the classic ASAM Transport Format (referenced as ATF/CLA and currently published as version 1.4.1). With version 5.0 comes the introduction of ATF/XML which is an XML-based version of the ATF file format. Like its older counterpart, ATF/XML makes it possible to exchange whole environments or even parts of those between different computers and environments. While there are slight differences other than the XML basis, ATF/XML and ATF/CLA are so similar that they should be considered to be two different “flavors” of the same standard.

ATF/XML is able to transport the required structure and instance information as well as the respective data and to store all this information in a standardized format.

This documentation describes and explains the ASAM Transport Format in XML (ATF/XML).

6.2 ATF/XML AND ATF/CLA COMPARISON

Both the ATF/CLA and ATF/XML specifications are based upon the ODS base model. This base model describes the base data types and more complex structures used in valid ODS application models. All ATF/CLA files and ATF/XML files use the components in the base model to sufficiently describe an application and measurement data such that the application can be understood by any ODS-aware processor. There are slight differences, however, in how the two formats define models and instance data.

The primary difference between ATF/CLA and ATF/XML is that ATF/CLA uses its own highly detailed specification for defining the structure of the file, while ATF/XML relies upon the W3C XML specification for defining the structure of the file. ATF/XML files must be well-formed, valid XML files and may be checked for structural validity by any XML validating parser. Indeed, it is this capability and the proliferation of XML processing tools that spurred the development of the ATF/XML file format. It is also possible to construct an XML schema file that, combined with a validating parser, can perform a more thorough validation of the ATF/XML file.

Another significant difference between the two flavors of ATF is that ATF/XML uses Unicode character encoding. The Unicode character set is a worldwide agreed-upon 16 bit character set, which implements a superset of the ASCII character set. ATF/CLA files can use the 256 ASCII characters, but due to different hardware/software configurations, only the first 128 are guaranteed to be supported. As a result, characters such as German umlauts can require escape characters for use in ATF/CLA. With Unicode, escape characters are not required.

There are minor differences between the two flavors of ATF. These differences include:

- In ATF/CLA, attributes derived from base attributes may declare a data type different than the base element from which they derived. ATF/XML does not allow any deviation from the base element.
- In ATF/CLA, application attributes and relations to other elements are defined in generally the same manner. ATF/XML delineates application attributes, instance attributes, and relational attributes.
- Both ATF/CLA and ATF/XML rely implicitly upon the base model. However, ATF/XML provides a base schema that can be used for basic validation and to be extended for more thorough validation.
- ATF/XML has an additional documentation section for specifying information about the file, how it was created, and when.
- In ATF/CLA files, lists and sequences of strings are delimited by commas. ATF/XML is hampered by a severe limitation of XML in that lists of values are delimited by spaces. This prevents a single value block from containing a sequence of strings. A sequence of value blocks is required.
- ATF/XML files are able to handle the latest extensions to the ODS base model – namely string length, obligatory flag, autogenerate flag, and unique flag. The extensions have not yet been added to the ATF/CLA specification.
- When specifying instance information in ATF/CLA files, the order of the instance elements and the order of attributes for each element are both arbitrary. By default in ATF/XML it is permissible to have arbitrary arrangements of elements and attributes, but it is also possible to create an external schema which rigidly enforces the order of elements and attributes.
- ATF/CLA allows in-line declaration of instance attributes merely by specifying the name of the attribute, its data type and its value. ATF/XML requires an explicit declaration of an instance attribute block containing all instance attribute elements. Each instance attribute element is declared with its name, data type, and value. The mechanism for declaring data type is different from ATF/CLA and allows application schemas to be written that define which data types, if any, are allowed.
- ATF/CLA supplies its own definition of acceptable data types, ATF/XML utilizes the underlying XML schema data types as building blocks for more complex data types.
- ATF/CLA supplies its own rules for whitespace, ATF/XML utilizes the XML specification for whitespace.
- ATF/CLA supplies its own rules for comments, ATF/XML utilizes the XML specification for comments.
- ATF/CLA supplies its own rules for identifiers, ATF/XML utilizes the XML specification for element names and attribute tag names. This presents a problem because the XML specification specifically excludes spaces from tag names. Full backward compatibility with existing models utilizing spaces in their tag names is not possible.
- ATF/CLA supplies its own rules for defining external files for storage of measurement data and for inclusion into the document. ATF/XML uses similar rules for defining external files for storage of measurement data, but utilizes the XML Inclusion (XInclude) specification for inclusion of other files into the XML document.

- ATF/CLA allows in-line declaration of user-defined enumerations. This is efficient if the enumeration is to be used only once, but inefficient if multiple elements are to re-use the enumeration. ATF/XML allows user-defined enumerations to be defined as separate elements that can be referred to by 1-n application elements.
- ATF/CLA allows a zero-based index integer representation of user-defined enumerations. This allows instance elements to merely specify the index of the appropriate enumeration item. ATF/XML does not allow such integer representation. Instance elements must explicitly state the desired enumerated value.
- ATF/XML allows the inverse relationship name to be specified when defining relations between element.
- ATF/XML allows the locale of the document to be specified. This provides information relevant to a processing application for displaying numbers and dates.
- ATF/XML allows relationships between elements to be defined such that xPointer-aware XML parsers can enforce the validity of the references to other elements. In ATF/CLA, only ASAM ODS-aware applications can enforce the validity of the references.

It is not a coincidence that that ATF/CLA and ATF/XML have so much in common. Rather than having two separate file transport formats, ATF/CLA and ATF/XML should be thought of as two different “flavors” of the same format. A person familiar with ATF/CLA will instantly recognize the similarities of ATF/XML. Consider the following example of the same instance element defined in ATF/XML and ATF/CLA.

EXAMPLE: **COMPARISON BETWEEN ATF/CLA AND ATF/XML**

The same instance is represented similarly in ATF/CLA and ATF/XML.

ATF/XML	ATF/CLA
`<Engine>`	`INSTELEM Engine`
`  <EngineName>Test Engine 1</EngineName>`	`EngineName = "TestEngine 1";`
`  <EngineId>1</EngineId>`	`EngineId = 1;`
`  <EngineVersion>A 1.0</EngineVersion>`	`EngineVersion = "A 1.0";`
`  <EngineDescription>The first test engine</EngineDescription>`	`EngineDescription = "The first test engine";`
`  <VersionDate></VersionDate>`	
`  <CreateDate>19971209141345</CreateDate>`	`CreateDate = "19971209141345";`
`  <EngineType>ABC47111</EngineType>`	`EngineType = "ABC4711";`
`  <bore>92.</bore>`	`bore = 92.;`
`  <cylindernumber>6</cylindernumber>`	`cylindernumber = 6;`
`  <stroke>95.3</stroke>`	`stroke = 95.3;`
`  <SubTests>1</SubTests>`	`SubTests = 1;`
`  <EngineUser>21</EngineUser>`	`EngineUser = 21;`
`</Engine>`	`ENDINSTELEM;`

6.3 STRUCTURE OF AN ATF/XML FILE

An ATF/XML file is a logical XML file consisting of one or more well-formed physical XML files. ATF/XML files may contain measurement data or pointers to external files containing the measurement data. The logical ATF/XML file is self-describing meaning that the application model is contained in the file along with the instance data.

With respect to the many different languages commonly used for defining application models and instance data, the encoding for the XML files shall be the Unicode format “UTF-8”.

The ATF file consists of a sequence of documentation, locale, base model info, description of external file components, application model meta-data, and instance data. Of these, only the base model info and the application model meta-data are required.

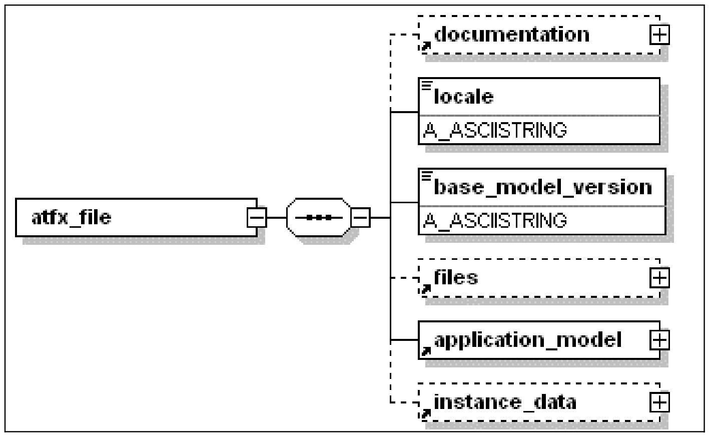


Figure 1 Structure of an ATF/XML File

6.4 LOGICAL AND PHYSICAL ATF/XML FILES

Logical ATF Files can be composed of multiple physical files. ATF/XML files and their classic ATF/CLA counterparts provide the opportunity to be self contained as well as the possibility to be split into multiple physical files that are re-assembled by the processing application into one logical file. This capability allows ATF designers to split the files into reusable modular libraries for consistency of commonly used components. It also allows large amounts of measurement data to stay in native format files.

The usage of multiple physical files in ATF/XML is slightly different than that of ATF/CLA files. There are two different mechanisms for including physical files. The method to use depends upon the type of information contained in the physical file. If the physical file contains snippets of valid xml, the XML XInclude is used and the XML parser automatically imports the file at the time of processing. If the physical file contains binary or other measurement data, the XML XInclude cannot be used and the processing application must obtain the file itself.

6.4.1 USING XINCLUDE TO COMBINE MULTIPLE XML DOCUMENTS

XInclude is a recent W3C specification for building large XML documents out of multiple well-formed XML documents, independently of validation. Each piece can be a complete XML document or a fragmentary XML document. This capability is ideally suited for ATF/XML developers that wish to develop a standard library of XML fragments that are included for use in many XML files. Typically this would be to have the application meta-data defined in one or more XML files that are included

To enhance reuse and modularity, XInclude is a technique for constructing new XML documents from existing ones by using a simple inclusion mechanism. Notes on using XInclude:

- http://www.w3.org/2001/XInclude is the official XInclude namespace
- XInclude is a very recent addition to the XML standard. It has not yet been finalized by the W3C and is not yet widely supported by XML tool suppliers.

EXAMPLE: **XINCLUDE**

Given an ATF/XML file named ‘xitest.xml’.
Given another ATF/XML file named ‘measurement.xml’
The result is identical to what would be taken from an ATF/XML file named ‘result.xml’ (the parser will combine those two files appropriately)

Content of ‘xitest.xml’:

```
<atfx_file
   version="atfx_file v1.0.1"
   xmlns="http://www.asam.net"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xi="http://www.w3.org/2001/XInclude"
   xsi:schemaLocation="http://www.asam.net odsbase_27_0_1.xsd">
   <application_model>
      <application_element>
         <xi:include href="library/measurement.xml" />
      </application_element>
   </application_model>
   <instance_data>
   ...
   </instance_data>
</atfx_file>
```

Content of ‘measurement.xml’:

```
<name>Measurement</name>
<basetype>AoMeasurement</basetype>
<application_attribute>
   <name>MeaName</name>
   <baseattribute>name</baseattribute>
   <unique>true</unique>
   <length>50</length>
</application_attribute>
<application_attribute>
   <name>MeaId</name>
   <baseattribute>id</baseattribute>
   <autogenerate>true</autogenerate>
   <obligatory>true</obligatory>
</application_attribute>
...
<relation_attribute>
      <name>MeaQuantities</name>
   <ref_to>MeasurmentQuantity</ref_to>
   <base_relation>measurement_quantities</base_relation>
   <min_occurs>0</min_occurs>
   <max_occurs>1</max_occurs>
</relation_attribute>
```

Content of ‘result.xml’:

```
<atfx_file version="atfx_file v1.0.1"
   xmlns="http://www.asam.net"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xi="http://www.w3.org/2001/XInclude"
   xsi:schemaLocation="http://www.asam.net odsbase_27_0_1.xsd">
   <application_model>
      <application_element>
         <name>Measurement</name>
         <basetype>AoMeasurement</basetype>
         <application_attribute>
            <name>MeaName</name>
            <baseattribute>name</baseattribute>
            <unique>true</unique>
            <length>50</length>
         </application_attribute>
         <application_attribute>
            <name>MeaId</name>
            <baseattribute>id</baseattribute>
            <autogenerate>true</autogenerate>
            <obligatory>true</obligatory>
         </application_attribute>
         ...
         <relation_attribute>
            <name>MeaQuantities</name>
            <ref_to>MeasurmentQuantity</ref_to>
            <base_relation>measurement_quantities</base_relation>
            <min_occurs>0</min_occurs>
            <max_occurs>1</max_occurs>
         </relation_attribute>
      </application_element>
   </application_model>
   <instance_data>
   ...
   </instance_data>
</atfx_file>
```

6.4.2 DECLARING THE ROOT ELEMENT

The root element of the XML file is the container for all other elements. The root element describes the version of the XML schema file that it adheres to and declares the namespaces used in the document.

As per the ASAM XML style guide, the version of the schema file is given in the version attribute. The format of the attribute is:

```
Schema-name: V<Application Profile Nr>.<Version Nr>.<Revision Nr>[<Patch Level>].
```

➢ Application Profile Number	(numeric: [0-9]+)
➢ Version Number	(numeric: [0-9]+)
➢ Revision Number	(numeric: [0-9]+)
➢ Patch level	(literal notation: [a-z]+)

```
<atfx_file  version="atfx_file: v1.0.1"
```

```
xmlns="http://www.asam.net"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation="http://www.asam.net odsbase_27_0_3.xsd">
```

6.4.3 DECLARING THE LOCALE

In order to support Internationalization, the locale of the document can be declared. This can be utilized by client applications for the purpose of formatting messages, numbers, currency and dates. The locale consists of international standard 2-character abbreviations for language (ISO 639) and country (ISO 3166). Either parts of a locale may be empty and are separated by underscore characters (`_'). Examples of locale names might include fr (French), de_CH (Swiss German)

EXAMPLE: **A LOCALE**

This locale specifies English as language and the USA as country.

```
<!--
***********************************************************
*      Locale of Document
***********************************************************
-->
      <locale>en_us</locale>
```

6.4.4 DECLARING THE ODS BASE MODEL

The ATF/XML file must declare the ODS base model upon which the application model is based.

EXAMPLE: **A BASE MODEL VERSION**

This code segment specifies ‘asam27’ as the version of the underlying ASAM ODS base model.

```
<!--
***********************************************************
*      Based on ODS Base Model version 27
***********************************************************
-->
      <base_model_version>asam27</base_model_version>
```

6.4.5 USING EXTERNAL FILES FOR MEASUREMENT DATA

External files may be used to hold measurement data as long as the files are declared in the files portion of the file and the local columns provide processing instructions for deciphering the files. The files portion of the ATF/XML file is identified by the <files> XML tag.

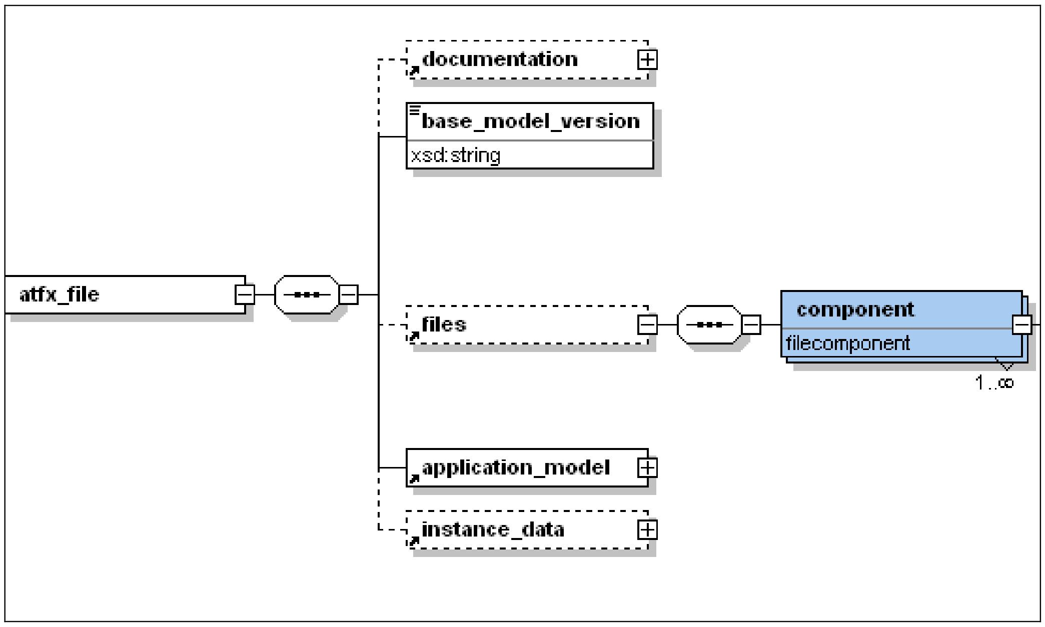


Figure 2: External File Component Placement

EXAMPLE: **AN EXTERNAL FILE DECLARATION**

This code extract from an XML file specifies a file named ‘k1.dat’ as external file. That file is placed in subdirectory ‘data’ which is parallel to the directory where the current XML file resides and which can be accessed from one level above. This file may be referenced as ‘file1’ furtheron in the XML file.

```
<atfx_file version="atfx_file v1.0.1"
   xmlns="http://www.asam.net"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.asam.net odsbase_27_0_1.xsd">
...
<!--
**************************************************************************
* Based on ODS Base Model version 27
**************************************************************************
-->
<base_model_version>asam27</base_model_version>

<!--
**************************************************************************
* declare any external files
**************************************************************************
-->
<files>
   <component>
      <identifier>file1</identifier>
      <filename>../data/k1.dat</filename>
   </component>
</files>
...
```

The path- and filenames used as references in an ATF/XML file must be transportable and usable on a wide range of computer architectures. Thus, a general format for the definition of path- and filenames is paramount, which has to be easily interpreted and converted for the target machine.

The Uniform Resource Locator (URL) used in the worldwide Internet is already working on different platforms. Through the naming conventions of the URL it will be possible for future ATF versions to have ATF files distributed within a network.

The URL describes a protocol for the target server, the target system the machine or server name, the port, the directory path and the filename. The URL has the following structure:

access_method://server_name[:port]/directory_path/filename

EXAMPLE: **AN URL**

This example shows an URL for a resource on a remote machine (the master list of all World Wide Web servers).
With the protocol HTTP (Web) the server named “www.w3.org” is addressed. By using the standard port the directory “/hypertext/DataSources/WWW” can be accessed, which contains the hypertext document “Geographical.html”.

```
http://www.w3.org/hypertext/DataSources/WWW/Geographical.html
```

Each file in the Internet is by means of the URL uniquely addressable.

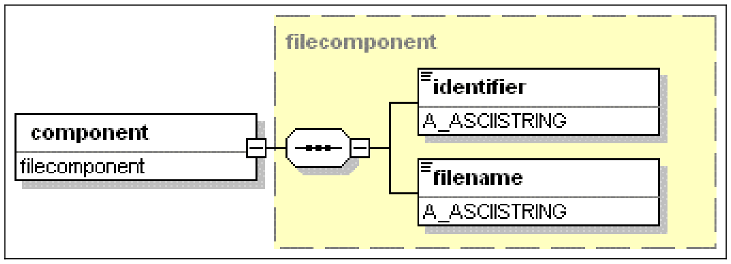


Figure 3: External File Component Structure

If the protocol references the contents of a directory, the URL ends with a slash „/“. When referencing a file within the directory no slash follows.

URLs as the ones described above are called absolute URLs, because they contain the full pathname of a file. A simplified format of URLs is called relative URL. This simplified format references other documents on the same server as the actual document. With relative URLs, an ATF file can reference directly the physical components without referencing the server.

Absolute URLs may reference any resource within the Internet, including local resources. Due to reasons explained later, relative URLs are better suited for local resources.

A relative URL implies the same access method, the same server name and the same directory path as the ATF file containing the URL. It describes the relative position of the component of the ATF file in relation to the actual ATF file.

Example:*../../matrix_xyz.dat*

Relative URLs make it possible to move the ATF directory structure to any position in the directory tree. A relative URL does not have to reference the directory of the actual document but may contain a reference relative to the root of the ATF file. This variant of relative URLs starts with a slash (/). It looks similar to an absolute URL but does not contain the access method and the server name.

The correct conversion of pathnames in both directions and the insertion of correct separators are the responsibility of the ATF reading and writing programs on the respective target systems.

If filenames are too long for the target system (e.g. length of DOS-filenames 8.3), the filenames will be shortened following these rules: After the first point within the filename a maximum of 3 more characters are allowed. None of these characters may be a point. If there are more than 3 characters or another point, the rest will be truncated. Unallowed characters are substituted by a tilde (swung dash).

The number of characters in front of the point is counted and truncated to 8 characters. If this results in filenames which are not unique, the algorithms from Windows 95 can be used (truncating to 6 characters and using the latter 2 characters for numbering).

To make the transfer of files to the world of UNIX easier, filenames will always be written in lowercase, even on operating systems which are not case-sensitive (VAX/VMS, MSWindows).

Filenames in uppercase are only used on demand when path and filenames on the UNIX machine should be in uppercase.

6.5 VALIDATING AN ATF/XML FILE

One of the benefits of using ATF/XML is the built-in validation capabilities of XML and XML Schema. However, this validation can only ensure that the file is well-formed and syntactically correct. It cannot validate all semantics of the file.

The ASAM ODS ATF/XML specification includes the ODS base schema file that can be used for validating the syntax of the application model portion of the file. This schema ensures that the model defined in the application model portion is a syntactically correct model. The ODS base schema file does not perform any validation of instance data.

Because application models can and will vary from application to application, the generic ODS base schema is insufficient for validating the syntax of the instance data portion of the ATF/XML file. Accordingly, an application specific schema may be used to validate the entire file. Such a schema would take into consideration the particular features of the given application model.

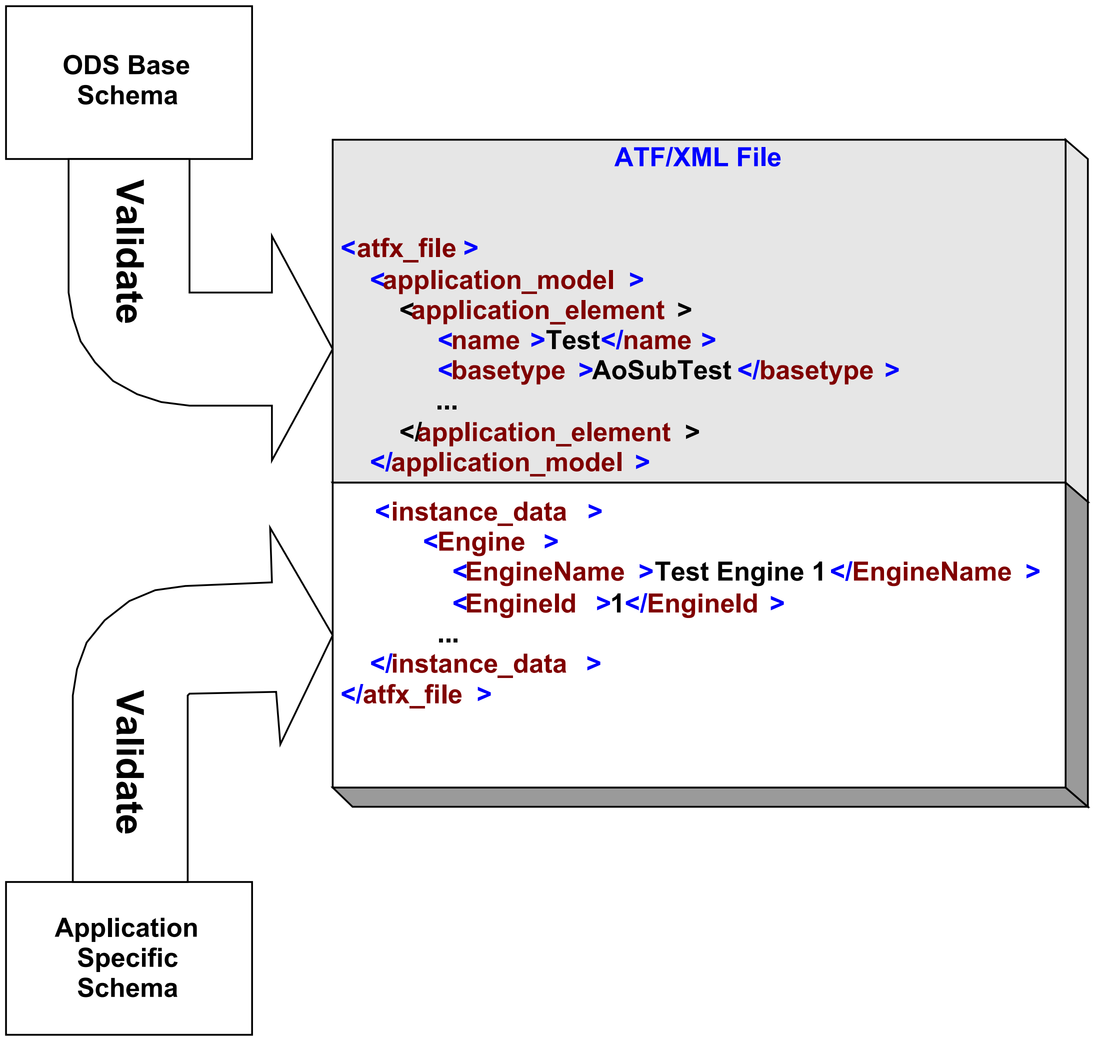


Figure 4: Using Schemas to Validate Files

An application specific schema would need to define the application model using XML Schema language (XSL).

EXAMPLE: **AN APPLICATION ELEMENT DEFINED USING XSL**

This application element is an extension of the ods:AoTest type defined in the ODS base schema. Each of the attributes for the element is defined along with the attributes' base types. This amounts to a redundant definition of the model using the XML Schema Definition language (XSD).

```
<xsd:complexType name="Engine">
 <xsd:complexContent>
  <xsd:extension base="ods:AoTest">
   <xsd:sequence>
    <xsd:element name="EngineName" type="ods:name"/>
    <xsd:element name="EngineId" type="ods:id"/>
    <xsd:element name="EngineVersion" type="ods:version" nillable="true"/>
    <xsd:element name="EngineDescription" type="ods:description"
            nillable="true"/>
    <xsd:element name="version_date" type="ods:version_date"
            nillable="true"/>
    <xsd:element name="CreateDate" type="ods:DT_DATE" nillable="true"/>
    <xsd:element name="EngineType" type="ods:A_ASCIISTRING"
            nillable="true"/>
    <xsd:element name="bore" type="ods:A_FLOAT32" nillable="true"/>
    <xsd:element name="cylindernumber" type="ods:A_INT16" nillable="true"/>
    <xsd:element name="stroke" type="ods:A_FLOAT32" nillable="true"/>
    <xsd:element name="SubTests" type="ods:id" nillable="true" />
    <xsd:element name="EngineUser" type="ods:id" nillable="true"/>
   </xsd:sequence>
  </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>
```

This definition of the model in XSD is required if it is desired to use a validating parser for validating the contents of the ATF/XML file.

Once the application-specific schema has been developed, it can be used separately to validate an ATF/XML file. While this step is technically unnecessary, it can provide another level assurance that the file meets the agreed-upon model. For example, if an OEM only wants to ensure that suppliers are providing syntactically correct ATF/XML files, they may require the suppliers validate the files using only the ODS base schema file. If, however, the OEM wants to require a more thorough validation, it may require the suppliers to validate using a more thorough, application-specific schema.

6.5.2 CREATING AN APPLICATION SCHEMA

The ODS base schema can be used only for validating the data model portion of the XML file. As a result, it is not designed to be a stand-alone schema. It is designed to be included as part of an application schema. Application schemas must provide validation instructions for the instance data portion of the XML file. At a minimum, application schemas should include the ODS base schema and provide bare-bones validation for the instance data portion.

EXAMPLE: A BARE-BONES APLICATION SCHEMA

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="http://www.asam.net"
            xmlns="http://www.asam.net"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified">
   <xsd:include schemaLocation="asam_base_model.xsd"/>

   <!-- *** declare Bare-Bones Instance Data section with no validation of
            contents  ***   -->
   <xsd:element name="instance_data">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:any processContents="skip" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

While this application schema is technically correct, most organizations will probably desire a more thorough application schema.

Using the include functionality of the schema language, it is possible to build generic or corporation specific schemas that can be reused as components of application schemas. This allows standardization of schema definitions and promotes reuse.

EXAMPLE: A STANDARDIZED APPLICATION SCHEMA

```
<?xml version="1.0"?>
   <xsd:schema targetNamespace="http://www.asam.net"
       xmlns="http://www.asam.net"
       xmlns:xsd=http://www.w3.org/2001/XMLSchema
       elementFormDefault="qualified">
   <xsd:include schemaLocation="asam_base_model.xsd"/>
   <xsd:include schemaLocation="corporate_model.xsd"/>
   <xsd:include schemaLocation="department_model.xsd"/>
</xsd:schema>
```

6.6 DEFINING AN APPLICATION MODEL

The application model portion of the ATF/XML file is used to describe the application elements and their relationships to one another. It is identified by the <application_model> XML tag. The application model block consists of a sequence of 1-N application elements followed by 0-N application enumeration elements.

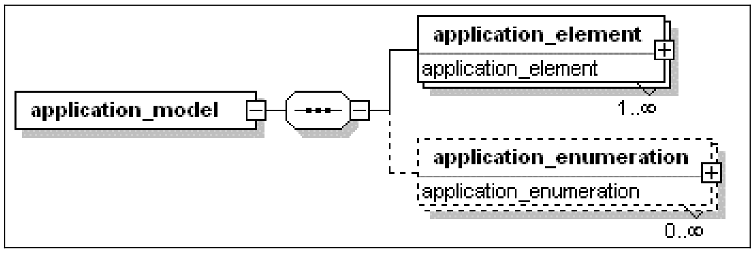


Figure 5: Application Model Components

EXAMPLE: AN APPLICATION MODEL DECLARATION

```
<!--
*******************************************************************
* declare application model meta data
*******************************************************************
-->
<application_model>
  <application_element>
    <!--      ***   declare Engine Element   ***       -->
    <name>Engine</name>
    ...
  </application_element>
</application_model>
```

Each of the application elements must be valid XML snippets that conform to the ODS base schema definition for application elements as well as conforming to the underlying ODS base model. This implies that the ODS base schema does not rigidly enforce all constraints in the ODS base model.

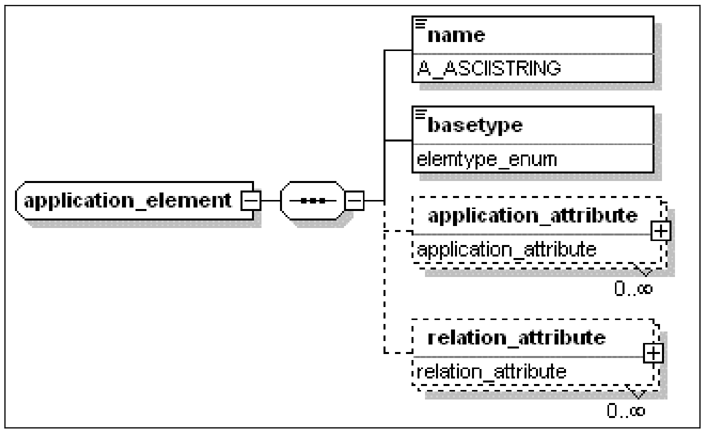


Figure 6: Application Element Structure

Application elements consist of the element name, basetype and a series of application_attribute elements followed by a series of relation_attribute elements. At a minimum, the application attributes specified must contain the required attributes specified in the ODS base model. Additional attributes are also allowed. Likewise, the relation attributes must specify any required relations for the given element type. Additional relationships may also be defined. Base attributes and relations that are declared as optional in the base model are optional in the ATF/XML file.

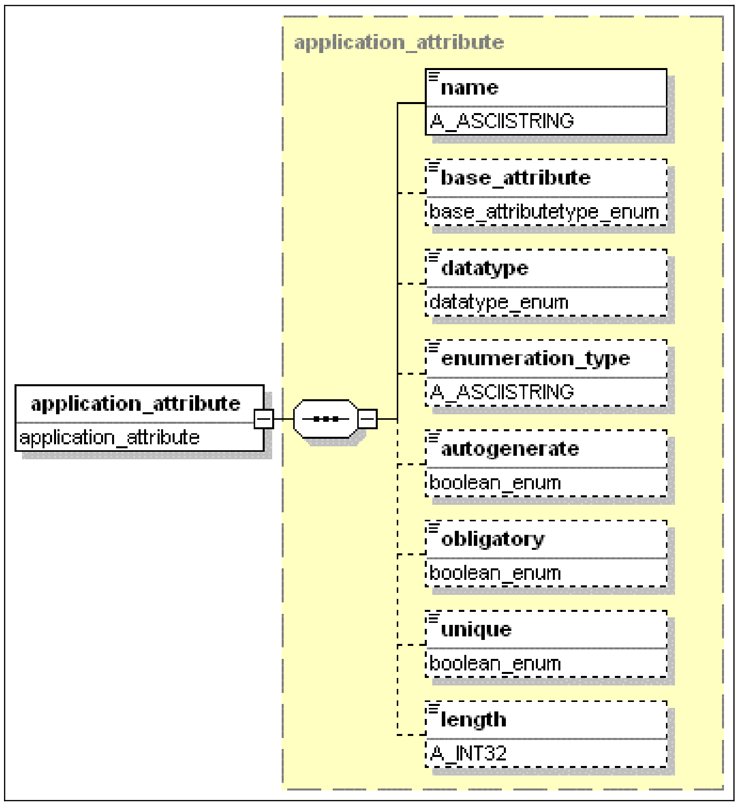


Figure 7: Application Attribute Structure

There are significant differences between ATF/CLA and ATF/XML for application attributes. ATF/XML files are able to handle the latest extensions to the ODS base model – namely string length, obligatory flag, autogenerate flag, and unique flag. The extensions have not yet been added to the ATF/CLA specification. Another difference is in declaring data types. In ATF/CLA, attributes derived from base attributes may declare a data type different than the base attribute from which they derived. ATF/XML does not allow any deviation from the base attribute.

Element Relationship Declaration

ATF/XML files allow explicit declaration of the relationships between elements. Relationships between parent elements and their children are quite common.

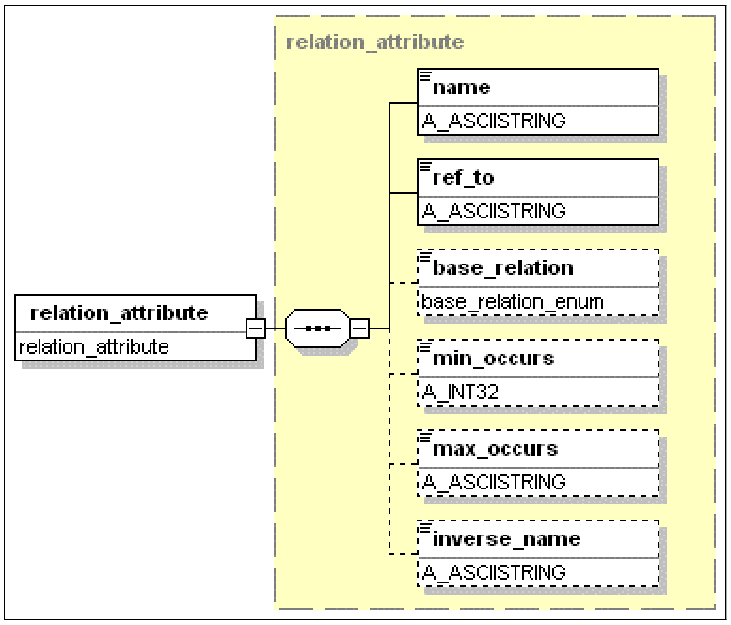


Figure 8: Relation Attribute Structure

Unlike in ATF/CLA, ATF/XML separates relation attributes from application attributes to provide a standard pattern for declaring relationships. Each declaration of a relationship must include the name of the relationship and the name of the related item. Relationships need not be based upon ODS base relations. If a relationship is needed in both directions, this can be modeled by explicitly specifying the name of the inverse relationship using the <inverse_name> tag.

EXAMPLE: A RELATION ATTRIBUTE DECLARATION

```
<relation_attribute>
   <name>TestUser</name>
   <ref_to>User</ref_to>
```

```
    <min_occurs>0</min_occurs>
    <max_occurs>1</max_occurs>
    <inverse_name>UserTests</inverse_name>
</relation_attribute>
```

Element Relationship Declaration in Application Schemas

It is possible to validate ATF/XML files using application-specific schemas. These schemas can be designed in such a manner as to enforce the referential integrity of the ATF/XML file. Using xPointer in the application schema allows an xPointer-aware tool to verify that the Ids referred to by elements are valid Ids. Of course, it is not always necessary to have the XML parser validate the file, but the option does exist.

To allow the ATF/XML file to have the links between elements validated, the links must be designed properly. Any element that is to be the target of a link must have a unique ID element that is declared using the xsd:Id declaration. Any element that links to the target element must have a link element that is declared using the xsd:anyURI declaration. The ODS base schema includes three helper types that may be used by application schemas.

```
<xsd:simpleType name="t_id">
  <xsd:restriction base="xsd:ID"/>
</xsd:simpleType>

<xsd:simpleType name="t_ref">
  <xsd:restriction base="xsd:anyURI"/>
</xsd:simpleType>

<xsd:simpleType name="t_ref_list">
  <xsd:list itemType="t_ref"/>
</xsd:simpleType>
```

The element t_id is used for declaring that the element is a unique identifier to be used as the target of a link.

The element t_ref is used for 1-1 relationships to declare that the element is any valid URI. In ATF/XML, the URI may either be the actual ID of the target element or an xPointer expression that points to the ID element of the target element.

The element t_ref_list is used for 1-N to declare that the element is a space-delimited series of valid URIs. In ATF/XML, the URIs may either be the actual IDs of the target elements or a series of xPointer expressions that point to the ID element of the target elements.

In the XML file, the xPointer is identified as an xPointer fragment using the #xPointer identifier followed by the xPointer expression that identifies the target of the link.

EXAMPLE: **AN XPOINTER**

In the first line of this example the link is to the element whose ID is 21. The second line shows the shortened form.

```
<EngineUser> #xPointer(id("21")) </EngineUser>
<EngineUser> #element(21)</EngineUser>
```

EXAMPLE:

PART OF AN APPLICATION SCHEMA USING XPOINTERS

The following snippet of an application schema is an example of using xPointers for verifying the relational integrity of the ATF/XML file. It declares that the <EngineId> element is a unique ID that can be the target of links from other elements. It also declares that the <SubTests> element is a list of xPointers to Ids of SubTest elements and that the <TestUser> element is an xPointer to the Id of a TestUser element.

The application schema:

```
<xsd:complexType name="Engine">
 <xsd:complexContent>
   <xsd:extension base="ods:AoTest">
    <xsd:sequence>
      <xsd:element name="EngineName" type="ods:name"/>
      <xsd:element name="EngineId" type="ods:t_id"/>
      <xsd:element name="EngineVersion" type="ods:version" nillable="true"/>
      <xsd:element name="EngineDescription" type="ods:description"
              nillable="true"/>
      <xsd:eement name="version_date" type="ods:version_date"
              nillable="true"/>
      <xsd:eement name="CreateDate" type="ods:DT_DATE" nillable="true"/>
      <xsd:element name="EngineType" type="ods:A_ASCIISTRING"
              nillable="true"/>
      <xsd:element name="bore" type="ods:A_FLOAT32" nillable="true"/>
      <xsd:element name="cylindernumber" type="ods:A_INT16" nillable="true"/>
      <xsd:element name="stroke" type="ods:A_FLOAT32" nillable="true"/>
      <xsd:element name="SubTests" type="ods:t_ref_list" nillable="true" />
      <xsd:element name="EngineUser" type="ods:t_ref" nillable="true"/>
    </xsd:sequence>
   </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>
```

A snippet from the corresponding ATF/XML file would look like this

```
<Engine>
   <EngineName>Test Engine 1</EngineName>
   <EngineId>16</EngineId>
   <EngineVersion>A 1.0</EngineVersion>
   <EngineDescription>The first test engine</EngineDescription>
   <version_date>199702010900</version_date>
   <CreateDate>199702010900</CreateDate>
   <EngineType>ABC47111</EngineType>
   <bore>92</bore>
   <cylindernumber>6</cylindernumber>
   <stroke>95.321</stroke>
   <SubTests>211 437 912 </SubTests>
   <EngineUser>21</EngineUser>
</Engine>
```

A snippet using xPointer would look like this

```
<SubTests>#xPointer(id("211"))  #xPointer(id("437"))  #xPointer(id("912"))
    </SubTests>
<EngineUser> #xPointer(id("21")) </EngineUser>.
```

EXAMPLE:

APPLICATION ELEMENT DECLARATION

```
<!--    ***  declare Test Element  ***  -->
<application_element>
   <name>Test</name>
   <basetype>AoSubTest</basetype>
   <application_attribute>
      <name>TestName</name>
```

```
        <baseattribute>name</baseattribute>
        <unique>true</unique>
        <length>50</length>
     </application_attribute>
     <application_attribute>
        <name>TestId</name>
        <baseattribute>id</baseattribute>
        <autogenerate>true</autogenerate>
        <obligatory>true</obligatory>
     </application_attribute>
     <application_attribute>
        <name>TestVersion</name>
        <baseattribute>version</baseattribute>
        <obligatory>true</obligatory>
     </application_attribute>
     <application_attribute>
        <name>TestDescription</name>
        <baseattribute>description</baseattribute>
        <length>100</length>
     </application_attribute>
     <application_attribute>
        <name>VersionDate</name>
        <baseattribute>version_date</baseattribute>
        <obligatory>false</obligatory>
     </application_attribute>
     <application_attribute>
        <name>CreateDate</name>
        <datatype>DT_DATE</datatype>
        <obligatory>false</obligatory>
     </application_attribute>
     <application_attribute>
        <name>TestType</name>
        <datatype> DT_STRING</datatype>
        <obligatory>false</obligatory>
     </application_attribute>
     <application_attribute>
        <name>TestComment</name>
        <datatype> DT_STRING</datatype>
        <obligatory>false</obligatory>
     </application_attribute>
     <application_attribute>
        <name>exhaust</name>
        <datatype> DT_STRING</datatype>
        <obligatory>false</obligatory>
     </application_attribute>
     <application_attribute>
        <name>air_filter</name>
        <datatype> DT_STRING</datatype>
        <obligatory>false</obligatory>
     </application_attribute>
     <!--      ***  related elements  *** -->
     <relation_attribute>
        <name>Measurements</name>
        <ref_to>Measurement</ref_to>
        <base_relation>children</base_relation>
        <min_occurs>0</min_occurs>
        <max_occurs>Many</max_occurs>
     </relation_attribute>
     <relation_attribute>
        <name>MainTest</name>
        <ref_to>Engine</ref_to>
        <base_relation>parent_test</base_relation>
        <min_occurs>0</min_occurs>
        <max_occurs>1</max_occurs>
     </relation_attribute>
     <relation_attribute>
        <name>TestUser</name>
        <ref_to>User</ref_to>
        <min_occurs>0</min_occurs>
        <max_occurs>1</max_occurs>
     </relation_attribute>
  </application_element>
```

For purposes of consistency, modularity, and re-usability, it is allowed to place the definitions of the application model into separate xml files. These files are included at run-time by any XInclude capable processor.

Application Enumeration Declaration

User-defined enumerations define the limited set of strings that are allowed as values for attribute values. In ATF/CLA, these values are declared in-line within the attribute's definition. In ATF/XML, user-defined enumerations are declared separately so that they may be reused by multiple attributes.

Each enumeration is defined using the <application_enumeration> tag. Within the tag, the name of the enumeration is given followed by enumeration item elements. Each item contains the name of the item and the item's value.

EXAMPLE: AN ENUMERATION

```
<application_enumeration>
  <name>MyEnum</name>
  <item>
     <name>first</name>
     <value>1</value>
  </item>
  <item>
     <name>second</name>
     <value>2</value>
  </item>
</application_enumeration>
```

Enumerations are used by application attributes. To declare that the values of an application attribute are limited to those of the user-defined enumeration, a link to that enumeration must be declared.

EXAMPLE: LINK TO AN ENUMERATION

```
<application_attribute>
   <name>air_filter</name>
   <datatype>DT_ENUM</datatype>
   <enumtype>MyEnum</enumtype>
</application_attribute>
```

6.7 USING SPACES IN ELEMENT & ATTRIBUTE NAMES

In the XML 1.0 Specification, element and attribute names may contain letters, digits, and assorted non-blank characters. Embedded spaces are not allowed. As a result, spaces are not allowed in ATF/XML element or attribute names. While it is possible to use escape characters within the content, the only escape characters you are allowed to use in element names are &amp, &apos, &gt, &lt, &quot. As a result, element names such as "measurement description" are not allowed.

Illegal syntax:

```
<Engine>
  <Engine Name>Test Engine 1</Engine Name>
  <Engine Id>1</Engine Id>
</Engine>
```

Full backward compatibility of existing data models utilizing spaces in the identifier names is not possible. It is suggested in the ASAM XML Styleguide that Hyphens be used in this instance.

Legal syntax:

```
<Engine>
  <Engine-Name>Test Engine 1</Engine-Name>
  <Engine-Id>1</Engine-Id>
</Engine>
```

6.8 DECLARING INSTANCE DATA

The instance data portion of the ATF/XML file is used to define instance elements. This portion of the ATF/XML file is identified by the <instance_data> XML tag. The instance data portion follows the application model portion and it is optional.

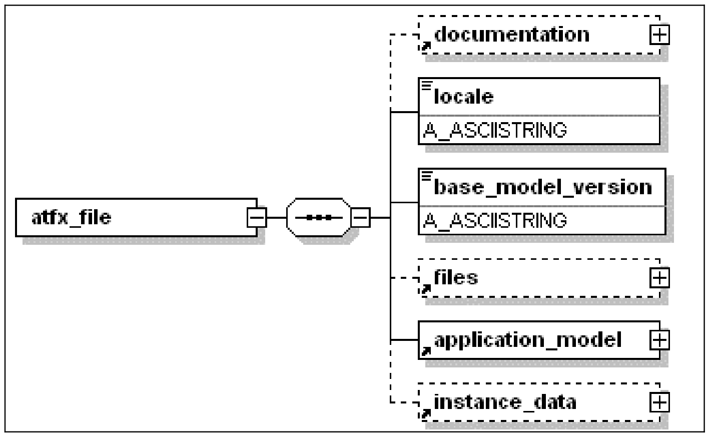


Figure 9: Instance Data Location in ATF/XML File

Each instance element is separated from each other instance element by XML tags with the instance element's name. Within the opening and closing tags for the instance element, a series of attribute value elements are given. These attribute values must match the definition of the corresponding application element. Defined attributes without assigned values are assumed to have the default value of UNDEFINED. When specifying instance information in ATF/XML files, the order of the instance elements and the order of attributes for each element are both arbitrary

Instance information is not validated by the ODS base schema because the structure of the instance data is a combination of the specific application model and the implicit ODS base model. It is possible, however, to create an external schema which rigidly enforces the order and contents of elements and attributes.

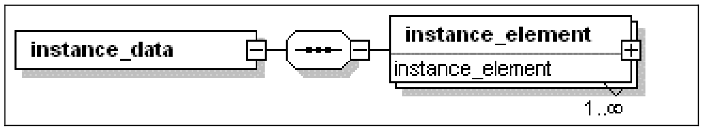


Figure 10: Instance Data Structure

The instance data section of the ATF/XML file is identified by the <instance_data> XML tags. The instance data section consists of a series of 0-N instance element declarations. Each instance element declaration is identified as an XML tag bearing the name of the instance element.

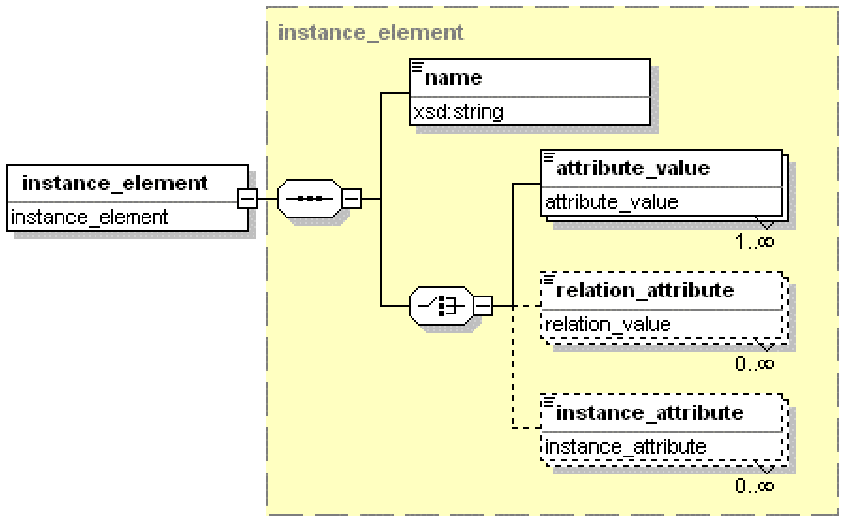


Figure 11: Instance Element Structure

Each instance element consists of an XML tag bearing the name of the instance element. Within the element is a series of 1-N attribute values followed by 0-N relation attributes and 0-N instance attributes. Each attribute value and relation attribute is an XML element where the XML tag consists of the attribute name. Inside the tag is the attribute's value(s).

EXAMPLE: **INSTANCE DATA**

This is a simple example of instance data

```
<instance_data>
   <Engine>
      <EngineName>Test Engine 1</EngineName>
      <EngineId>1</EngineId>
      <EngineVersion>A 1.0</EngineVersion>
      <EngineDescription>The first test engine</EngineDescription>
      <Mfg_Plant>
         <s>Ypsilanti</s>
         <s>Stockholm</s>
         <s>Turin</s>
         <s>San Francisco</s>
      </Mfg_Plant>
      <VersionDate>199702010900</VersionDate>
      <CreateDate>199702010900</CreateDate>
      <EngineType>ABC47111</EngineType>
      <bore>92</bore>
      <cylindernumber>6</cylindernumber>
      <stroke>95.321</stroke>
      <SubTests>1</SubTests>
      <EngineUser>21 22 23 24 25 87 62 </EngineUser>
   </Engine>
   ...
</instance_data>
```

Regardless of the data type, attributes with single values shall place the value within the attribute's tags. For attributes with multiple values e.g. lists, the data type determines how the values are declared. This is due to how XML handles lists and sets of data. XML 1.0 declares that lists of values are to be delimited by spaces. As a result, string values with embedded spaces will be treated differently than string values without embedded spaces. For the ATF/XML designer, this means that multiple occurrences of string data types shall be handled differently than non-string data types. Attributes with multiple values e.g. lists and sets, shall place non-string values as a series of 1-n, space-delimited values within the attribute's tags. Multiple string values shall be declared using a single string-sequence element with 1-n child elements, each containing a single string value.

EXAMPLE: **ATTRIBUTE WITH MULTIPLE VALUES**

This is an example of an attribute with multiple values (non-string)

```
<EngineUser>21 22 23 24 25 87 62 </EngineUser>
```

EXAMPLE: **ATTRIBUTE WITH MULTIPLE VALUES**

This is an example of an attribute with multiple values (string)

```
<Mfg_Plant>
    <s>Ypsilanti</s>
    <s>Stockholm</s>
    <s>Turin</s>
    <s>San Francisco</s>
</Mfg_Plant>
```

Base Schema Definition of String Sequence

```
<xsd:complexType name="string_sequence">
   <xsd:sequence minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="s" type="A_ASCIISTRING"/>
   </xsd:sequence>
</xsd:complexType>
```

Application Schema Using String Sequence

```
<xsd:element name="Mfg_Plant" type="string_sequence"/>
```

6.8.2 DECLARING INSTANCE ATTRIBUTES

Instance attributes are name-datatype-value triplets that are declared in-line within the declaration of an instance element. This differs from other attribute values whose names & data types are defined in the application model.

EXAMPLE: INSTANCE ELEMENT CONTAINING INSTANCE ATTRIBUTES

```
<User>
  <UserName>Todd Martin</UserName>
  <UserId>22</UserId>
  <UserVersion>1B</UserVersion>
  <UserDescription>Sub user group 1b</UserDescription>
  <version_date>199708101105</version_date>
  <password>j</password>
  <groups>30 40</groups>
  <UserDepartment>OP/ABC</UserDepartment>
  <AliasName>TM</AliasName>
  <instance_attributes>
     <inst_attr_A_INT32 name="altkey">123</inst_attr_A_INT_32>
     <inst_attr_A_ASCIISTRING name="passkey">hello</inst_attr_A_ASCIISTRING>
  </instance_attributes>
</User>
```

Instance attributes are grouped together within the <instance_attributes> tags. There should be at least one instance attribute within the tags or the optional tags should be omitted. Each instance attribute is declared using the appropriate instance attribute element. The instance attribute elements are predefined in the ODS base schema. The name of the instance attribute element describes the data type that the attribute is allowed to contain. By declaring that instance attributes of a given type are allowed, the XML validating parser can ensure that the values match the specified type.

The name of the instance element is declared as an attribute and the value of the attribute is declared between the attribute's tags. In the example above, the instance element has an instance attribute named "altkey", with a data type of long, and a value of 123.

By default, the ODS base schema does not perform any validation on instance data. Instance attributes were designed to allow application schemas to be written to perform validation on the instance data. It is possible to write a schema that limits the types of instance data allowed. It is also possible to create application schemas that do not allow any instance attributes to be used.

6.8.3 USING EXTERNAL FILE COMPONENTS FOR MEASUREMENT DATA

In ASAM-ODS the measured data is described by the base element *Local Column*. These columns are referred by measurement quantities and submatrices. While other elements like AoMeasurementQuantity and AoSubmatrix can be dealt with as normal base elements, the local column often refers large external binary datasets. Mainly for this reason, the COMPONENT construct was provided in both the ATF and ATF/XML specifications. This construct allows large amounts of binary data to be stored outside the XML file and therefore can save considerable time and space in transmission and storage.

Each external file component must have been defined within the <files> block. Once defined, a reference to the file shall be inserted where the binary component is to be inserted. The component description describes how the values are to be obtained from the external file for usage as if they had been stored locally.

EXAMPLE: **EXTERNAL FILE COMPONENT DECLARATION**

```
<files>
  <component>
    <identifier> binary_file_1</identifier>
    <filename>../data/k1.dat</filename>
  </component>
</files>
```

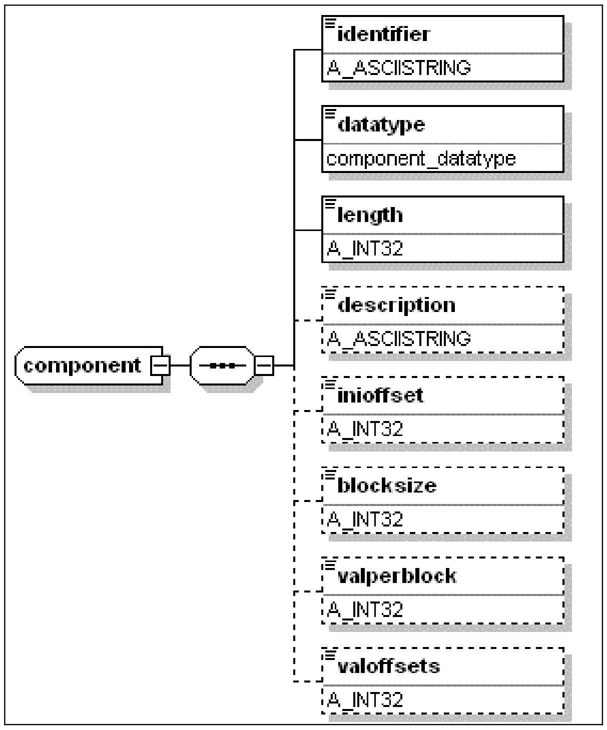


Figure 12: Structure of Component

EXAMPLE: **COMPONENT DECLARATION**

This is a simple example of a component declaration

```
<component>
   <identifier>binary_file_1</identifier>
   <datatype>A_FLOAT64</datatype>
   <length>11</length>
   <description>double precision floating-point numbers </description>
   <inioffset>0</inioffset>
   <blocksize>8</blocksize>
   <valperblock>1</valperblock>
   <valoffsets>0</valoffsets>
</component>
```

Within the component XML tags, a series of elements describes the values contained in the component and how they can be used.

The first element is the <identifier>. The value of the identifier must be one of the components listed in the FILES block at the beginning of the ATF/XML file.

The <datatype> element specifies the manner in which the data is stored on the file. In ATF/XML, the following ASAM datatypes are accepted:

- A_ASCIISTRING
- A_BYTEFIELD
- A_BITFIELD
- A_BOOLEAN
- A_FLOAT32
- A_FLOAT64
- A_FLOAT32_BEO
- A_FLOAT64_BEO
- A_INT8
- A_INT16
- A_INT32
- A_INT64
- A_INT16_BEO
- A_INT32_BEO
- A_INT64_BEO
- A_UINT8
- A_UINT16
- A_UINT32
- A_UINT64

The <length> specification element is an integer value that specifies the amount of data to be read from the file. In case of A_ASCIISTRING, DT_BYTESTR and A_BYTEFIELD it denotes the number of bytes to be read; in all other cases (integers, floats, bits and bytes) it denotes the number of values to be read. In the case of blobs this specification is equivalent with the length of the file on which the blob is located, because in a <component> only one blob can be referred, and only one blob may reside in a component file.

All other types may occur in sequences on a component file.

The <inioffset> element specifies the number of bytes in the file header. This file header is skipped when accessing the file. Since a blob always covers a whole file, there is no INIOFFSET in blob components.

The <blocksize> element is an integer that contains the size of each block in bytes. After the header, the main portion of the file should contain 1-n blocks. A block consists of values which may belong to different components and (in case of numerical data) may have different data types. Within a block also several values may belong to the same component, see examples at the end of this section. For all numerical data types a blocksize must be specified.

The <valperblock> element contains the number of values in each block belonging to the desired component. For each of these values the offset within a block must be specified in bytes. All these offsets are given within the valoffsets element. The <valoffsets> element contains a list of 1-n space-delimited offsets. An example is shown in the examples below.

REPRESENTATION OF A_BOOLEAN

In a binary component file each boolean shall be represented by one bit (0=false, 1=true). These bits are filled into a sequence of bytes. If the number of bits is not sufficient to fill a byte completely, then the rest of that byte shall remain unused. The byte is filled from left to right, i.e. beginning with the most significant bit. In the byte sequence (by_1, by_2, by_3,...) the n^{th} Boolean (n is one-based) can be located using the following integer formula. Assuming the bit positions in a byte numbered (again from left to right) 7,6,5,4,3,2,1,0 then the n^{th} Boolean is in the lby^{th} byte at bit position lbi:

lby = (n+7) / 8

lbi = lby*8 – n

The number of used bytes (Nby) can be calculated from the number of Booleans (Nbi) by integer arithmetic

Nby = 1 + (Nbi-1) / 8.

The value within the <length> element specifies the number of boolean values that are coded; the number of bytes occupied in the component file may be calculated as above.

REPRESENTATION OF A_INT8

This data type is represented simply by bytes. Thus the value within the <length> element denotes the number of byte values in the sequence, in case of a single value it is set to 1.

REPRESENTATION OF A_INT16

This 16-bit integer data type is represented by 2 bytes. If the keyword A_INT16 is given in the <datatype> element, then the least significant byte comes first (lower address), the most significant byte comes last (higher address). If the keyword A_INT16_BEO is given, then the most significant byte comes first, the least significant byte comes last; this order is also known as "Big Endian Order". The value within the <length> element denotes the number of integer values in the sequence, in case of a single value it is set to 1.

REPRESENTATION OF A_INT32

This 32-bit integer data type is represented by 4 bytes. If the keyword A_INT32 is given in the <datatype> element, then the least significant byte comes first (lower address), the most significant byte comes last (higher address). If the keyword A_INT32_BEO is given, then the most significant byte comes first, the least significant byte comes last; this order is also known as "Big Endian Order". The value within the <length> element denotes the number of integer values in the sequence, in case of a single value it is set to 1.

REPRESENTATION OF A_INT64

This 64-bit integer data type is represented by 8 bytes. If the keyword A_INT64 is given in the <datatype> element, then the least significant byte comes first (lower address), the most significant byte comes last (higher address). If the keyword A_INT64_BEO is given, then the most significant byte comes first, the least significant byte comes last; this order is also known as "Big Endian Order". The value within the <length> element denotes the number of integer values in the sequence, in case of a single value it is set to 1.

REPRESENTATION OF A_FLOAT32

This data type is represented by 4 bytes. If the keyword A_FLOAT32 is given in the <datatype> element, then the least significant byte comes first (lower address), the most significant byte comes last (higher address). If the keyword A_FLOAT32_BEO is given, then the most significant byte comes first, the least significant byte comes last; this order is also known as "Big Endian Order". The location of sign, exponent and mantissa is described in IEEE 754. The value within the <length> element denotes the number of float values in the sequence, in case of a single value it is set to 1.

REPRESENTATION OF A_FLOAT64

This data type is represented by 8 bytes. If the keyword A_FLOAT64 is given, then the least significant byte comes first (lower address), the most significant byte comes last (higher address). If the keyword A_FLOAT64 _BEO is given, then the most significant byte comes first, the least significant byte comes last; this order is also known as "Big Endian Order". The location of sign, exponent and mantissa is described in IEEE 754. The value within the <length> element denotes the number of float values in the sequence, in case of a single value it is set to 1.

REPRESENTATION OF A_COMPLEX32 & A_COMPLEX64

Complex numbers are controlled by the keywords A_FLOAT32 and A_FLOAT64 (resp. A_FLOAT32_BEO and A_FLOAT64_BEO) specified in the <datatype> element. The number of values must be doubled (i.e. only even numbers are legal). For a single complex value the value within the <length> element is 2, for a sequence of 7 complex values the value is 14. The order in complex sequences is

- real part of first sequence member
- imaginary part of first sequence member
- real part of second sequence member
- imaginary part of second sequence member
- ...

REPRESENTATION OF A_ASCIISTRING

This data type is represented by bytes. Each string is terminated by a NULL byte which increases the required space for that string by 1. In a sequence of strings each member of the sequence has a terminating NULL byte, and these NULL bytes are included in the total length of the component which is given in the value within the <length> element. The data type A_ASCIISTRING may not be mixed with other data types on the same component file.

REPRESENTATION OF A_BITFIELD

An A_BITFIELD datastream is represented as multiple of bytes. In a bitfield the first 2 bytes denote the number of bits coded in the subsequent bytes. Eventually not-used bitpositions are set to zero.
In a sequence of bitfields the next length indication follows immediately after the last byte of the previous member of the sequence. Please note that the 2 length indication bytes are always given in “Big Endian Order”, i.e. with the most significant byte first.

EXAMPLE:

2 bitfields with contents "011010110" and "11001" are represented by
00 09 00 D6 00 05 19
The total length is 7, this 7 is given as value of the <length> element.

The data type A_BITFIELD must not be mixed with other data types on the same component file.

REPRESENTATION OF A_BYTEFIELD

This data type is represented by bytes. In a bytefield the first 4 bytes denote the number of bytes following after these 4 bytes. In a sequence of bytefields the next length indication follows immediately after the last byte of the previous member of the sequence. Please note that the 4 length indication bytes are always given in “Big Endian Order”, i.e. with the most significant byte first.

EXAMPLE:

2 bytefields with contents "ABCDEFG" and "XYZ" are represented by
00 00 00 07 41 42 43 44 45 46 47 00 00 00 03 58 59 5A
The total length is 18, this 18 is given as value of the <length> element.

The data type A_BYTEFIELD must not be mixed with other data types on the same component file.

The following examples show how measurement data may be stored using ATF/XML.

EXAMPLE 1: LOCAL COLUMN WITHOUT <COMPONENT>

```
<LocalColumn>
   <name>MD</name>
   <id>2</id>
   <version/>
   <description/>
   <version_date/>
   <flags>0</flags>
   <global_flag>0</global_flag>
   <independent>false</independent>
   <minimum>150</minimum>
   <maximum>255.</maximum>
   <datatype>A_FLOAT32</datatype>
   <values>
     150. 160. 170. 180. 190. 200. 210. 220 230. 240. 250. 255. 250. 230. 190.
   </values>
   <submatrix>1</submatrix>
   <measurement_quantity>2</measurement_quantity>
</LocalColumn>
```

EXAMPLE 2: ONE LOCAL COLUMN ON ONE FILE

This local column uses a file identified by binary_file_1, which must have been previously defined in the files section. The file contains one measurement quantity with one measured value channel with double precision floating-point numbers.

```
<LocalColumn>
   <name>MD</name>
   <id>2</id>
   <version/>
   <description/>
   <version_date/>
   <flags>0</flags>
   <global_flag>0</global_flag>
   <independent>false</independent>
   <minimum>150</minimum>
   <maximum>255.</maximum>
   <datatype>A_FLOAT32</datatype>
   <values>
      <component>
         <identifier>binary_file_1</identifier>
         <length>11</length>
         <description>double precision floating-point numbers </description>
         <inioffset>0</inioffset>
         <blocksize>8</blocksize>
         <valperblock>1</valperblock>
         <valoffsets>0</valoffsets>
      </component>
   </values>
   <submatrix>1</submatrix>
   <measurement_quantity>2</measurement_quantity>
</LocalColumn>
```

EXAMPLE 3: TWO LOCAL COLUMNS ON ONE FILE – ONE AFTER THE OTHER

These local columns use a file identified by binary_file_2, which must have been previously defined in the files section. The file contains two measurement quantities:

- One measured channel with 10 double precision floating-point values.
- One measured channel with 10 short integer values.

```
<LocalColumn>
   <name>MD</name>
   <id>2</id>
   <version/>
   <description/>
   <version_date/>
   <flags>0</flags>
   <global_flag>0</global_flag>
   <independent>false</independent>
   <minimum>150</minimum>
   <maximum>255.</maximum>
   <datatype>A_FLOAT32</datatype>
   <values>
      <component>
         <identifier>binary_file_2</identifier>
         <length>10</length>
         <description>double precision floating-point numbers </description>
         <inioffset>0</inioffset>
         <blocksize>8</blocksize>
         <valperblock>1</valperblock>
         <valoffsets>0</valoffsets>
      </component>
   </values>
   <submatrix>1</submatrix>
   <measurement_quantity>2</measurement_quantity>
</LocalColumn>
```

The second local column on the file follows the first local column. Because it follows the first column on the same file, its inioffset value is set at 80 to denote that the first column was 80 bytes in length. The second column is defined as:

```
<LocalColumn>
   <name>QS</name>
   <id>3</id>
   <version/>
   <description/>
   <version_date/>
   <flags>0</flags>
   <global_flag>0</global_flag>
   <independent>false</independent>
   <minimum>125</minimum>
   <maximum>2455.</maximum>
   <datatype>A_INT16</datatype>
   <values>
      <component>
         <identifier>binary_file_2</identifier>
         <length>10</length>
         <description>short integer numbers</description>
         <inioffset>80</inioffset>
         <blocksize>8</blocksize>
         <valperblock>1</valperblock>
         <valoffsets>0</valoffsets>
      </component>
   </values>
   <submatrix>1</submatrix>
   <measurement_quantity>2</measurement_quantity>
</LocalColumn>
```

EXAMPLE 4: **TWO LOCAL COLUMNS ON ONE FILE – ONE ALTERNATING WITH THE OTHER**

These local columns use a file identified by binary_file_2, which must have been previously defined in the files section. The file contains two measurement quantities which were measured with the same sampling rate:

- One generated time channel (1000 double precision floating-point values)
- One measured channel (1000 short integer values)

The values were written alternately to the file. They are referred to by two local columns.

```
<LocalColumn>
   <name>MQ</name>
   <id>6</id>
   <version/>
   <description/>
   <version_date/>
   <flags>0</flags>
   <global_flag>0</global_flag>
   <independent>false</independent>
   <minimum>150</minimum>
   <maximum>255.</maximum>
   <datatype>A_FLOAT32</datatype>
   <values>
      <component>
         <identifier>binary_file_2</identifier>
         <length>1000</length>
         <description>double precision floating-point numbers </description>
         <inioffset>0</inioffset>
         <blocksize>10</blocksize>
         <valperblock>1</valperblock>
         <valoffsets>0</valoffsets>
      </component>
   </values>
   <submatrix>1</submatrix>
   <measurement_quantity>2</measurement_quantity>
</LocalColumn>
```

The second column is defined as:

```
<LocalColumn>
   <name>QS</name>
   <id>3</id>
   <version/>
   <description/>
   <version_date/>
   <flags>0</flags>
   <global_flag>0</global_flag>
   <independent>false</independent>
   <minimum>125</minimum>
   <maximum>2455.</maximum>
   <datatype>A_INT16</datatype>
   <values>
      <component>
         <identifier>binary_file_2</identifier>
         <length>1000</length>
         <description>short integer numbers</description>
         <inioffset>0</inioffset>
         <blocksize>10</blocksize>
         <valperblock>1</valperblock>
         <valoffsets>8</valoffsets>
      </component>
   </values>
   <submatrix>1</submatrix>
   <measurement_quantity>2</measurement_quantity>
</LocalColumn>
```

EXAMPLE 5: **THREE LOCAL COLUMNS WITH DIFFERENT SAMPLING RATES ON ONE FILE**

These local columns use a file identified by binary_file_4, which must have been previously defined in the files section. The file contains three measurement quantities which were measured with different sampling rates:

- One measured value channel with double precision floating-point numbers @ 10 Hertz.
- One measured value channel with short integer numbers @ 20 Hertz.
- One measured value channel with long integer numbers @ 30 Hertz.

The values were written to the file as they came from the measuring instrument. Before the first measurement point the device sends a header with 32 Bytes of parameters.

The first local column is defined as:

```
<LocalColumn>
   <name>MQ</name>
   <id>8</id>
   <version/>
   <description/>
   <version_date/>
   <flags>0</flags>
   <global_flag>0</global_flag>
   <independent>false</independent>
   <minimum>150</minimum>
   <maximum>255.</maximum>
   <datatype>A_FLOAT32</datatype>
   <values>
      <component>
         <identifier>binary_file_5</identifier>
         <length>10000</length>
         <description>double precision floating-point numbers </description>
         <inioffset>32</inioffset>
         <blocksize>24</blocksize>
         <valperblock>1</valperblock>
         <valoffsets>0</valoffsets>
      </component>
   </values>
   <submatrix>1</submatrix>
   <measurement_quantity>2</measurement_quantity>
</LocalColumn>
```

The second column is defined as:

```
<LocalColumn>
    <name>QS</name>
    <id>9</id>
    <version/>
    <description/>
    <version_date/>
    <flags>0</flags>
    <global_flag>0</global_flag>
    <independent>false</independent>
    <minimum>125</minimum>
    <maximum>2455.</maximum>
    <datatype>A_INT16</datatype>
    <values>
      <component>
         <identifier>binary_file_5</identifier>
         <length>20000</length>
         <description>short integer numbers.</description>
         <inioffset>0</inioffset>
         <blocksize>10</blocksize>
         <valperblock>2</valperblock>
         <valoffsets>8 18</valoffsets>
      </component>
```

```
   </values>
   <submatrix>2</submatrix>
   <measurement_quantity>3</measurement_quantity>
</LocalColumn>
```

The third column is defined as:

```
<LocalColumn>
   <name>QM</name>
   <id>10</id>
   <version/>
   <description/>
   <version_date/>
   <flags>0</flags>
   <global_flag>0</global_flag>
   <independent>false</independent>
   <minimum>125</minimum>
   <maximum>2455.</maximum>
   <datatype>A_INT32</datatype>
   <values>
      <component>
         <identifier>binary_file_5</identifier>
         <length>30000</length>
         <description> short integer numbers.</description>
         <inioffset>32</inioffset>
         <blocksize>24</blocksize>
         <valperblock>3</valperblock>
         <valoffsets>10 14 20</valoffsets>
      </component>
   </values>
   <submatrix>3</submatrix>
   <measurement_quantity>4</measurement_quantity>
</LocalColumn>
```

6.9 SECURITY

6.9.1 STORING SECURITY INFORMATION

Security information is used to accompany archive data in order to be able to restore the data completely, i.e. together with the access rights. For exchange purposes it is necessary to write both ATF/CLA and ATF/XML files without security information. ATF/CLA and ATF/XML writers therefore need a switch to select either “with” or “without” security information. ATF readers need at least a defined default behavior if they encounter a file with security information; this behavior is either “use” or “ignore” security information.

- On export the writing of security information on ATF file is optional.
- On import the use of security information from the ATF file is optional.
- On reading a file without security information the local security information is not influenced.

WRITING SECURITY INFORMATION ON ATF

Security information consists of instances of entities that are subtypes of

- AoUser
- AoUsergroup
- ACL

The subtypes of AoUser and AoUsergroup are application elements while the subtypes of ACL are not subject to any influence from the user. Only the two predefined subtypes ACLA (used for an application element and –if an attribute name is specified- for the attribute of an application element) and ACLI (used for instance elements) are allowed. No special syntax is applied to the security information. It is written according to the same rules as all other data.

- The entities are written in the metadata section together with the other application elements.
- The instances are written in the data section together with the other instance elements.

EXAMPLE: USER & USERGROUP DECLARATIONS

```
<!--
**********************************************************************
*  instance elements of the application elements User, User Group
**********************************************************************
-->
<User>
   <UserName>Peter Sellers</UserName>
   <UserId>21</UserId>
   <UserVersion>1A</UserVersion>
   <UserDescription>Super user group 1</UserDescription>
   <version_date>199708101105</version_date>
   <password>^1kjws7hxoish^mjkkdjxo”e--@@@kjg798xh0880djdj90</password>
   <groups>30 50</groups>
   <UserDepartment>OP/EFG</UserDepartment>
   <AliasName>PS</AliasName>
</User>
<User>
   <UserName>Todd Martin</UserName>
   <UserId>22</UserId>
   <UserVersion>1B</UserVersion>
   <UserDescription>Sub user group 1b</UserDescription>
   <version_date>199708101105</version_date>
   <password>j</password>
   <groups>30 40</groups>
   <UserDepartment>OP/ABC</UserDepartment>
   <AliasName>TM</AliasName>
</User>
<User>
   <UserName>Otto Bierman</UserName>
   <UserId>23</UserId>
   <UserVersion>1A</UserVersion>
   <UserDescription>Sub user group 1</UserDescription>
   <version_date>199708101105</version_date>
   <password>89xzhuon3m,”))?imi/U(/%unpo39048ejdkádksos••o•3</password>
   <groups>40 50</groups>
   <UserDepartment>OP/EFG</UserDepartment>
   <AliasName>OB</AliasName>
</User>
<UserGroup>
   <GroupName>AQ 1</GroupName>
   <GroupId>30</GroupId>
   <GroupVersion>00</GroupVersion>
   <GroupDescription>Group1</GroupDescription>
   <version_date>199708101105</version_date>
   <superuser_flag>true</superuser_flag>
   <users>21 22</users>
</UserGroup>
<UserGroup>
   <GroupName>AQ 2</GroupName>
   <GroupId>40</GroupId>
   <GroupVersion>00</GroupVersion>
   <GroupDescription>Group 2</GroupDescription>
   <version_date>199708101105</version_date>
   <superuser_flag>false</superuser_flag>
   <users>22 23</users>
</UserGroup>
<UserGroup>
   <GroupName>TZU 23</GroupName>
   <GroupId>50</GroupId>
   <GroupVersion>00</GroupVersion>
   <GroupDescription>Group 3</GroupDescription>
   <version_date>199708101105</version_date>
   <superuser_flag>false</superuser_flag>
   <users>21 23</users>
</UserGroup>
```

EXAMPLE: **ATTRIBUTE-LEVEL SECURITY**

```
<!--
***************************************************************************
*     Access Control List information for enforcing security
***************************************************************************
-->
<ACLA>
   <users>30</users>
   <appl_element_id>333</appl_element_id>
   <rights>7</rights>
   <!--- protection of whole application element -->
   <attribute_name/>
</ACLA>
<ACLA>
   <users>30</users>
   <appl_element_id>333</appl_element_id>
   <rights>3</rights>
   <!--- protection of description attribute -->
   <attribute_name>description</attribute_name>
</ACLA>
<ACLA>
   <users>40</users>
   <appl_element_id>123</appl_element_id>
   <rights>10</rights>
   <!--- protection of whole application element -->
   <attribute_name/>
</ACLA>
```

EXAMPLE: **INSTANCE-LEVEL SECURITY**

```
<ACLI>
   <users>50</users>
   <appl_element_id>123</appl_element_id>
   <rights>10</rights>
   <!--- protection of instance element -->
   <instance_id>23</instance_id>
</ACLI>
<ACLI>
   <users>40</users>
   <appl_element_id>123</appl_element_id>
   <rights>10</rights>
   <!--- protection of instance element -->
   <instance_id>23</instance_id>
</ACLI>
```

6.10 DATA TYPE USAGE IN ATF/XML

Some general information on data types used by ASAM ODS can be found in chapter 2.5.

As ATF/XML is a new standard, introduced with ASAM ODS version 5.0, it will use the ASAM data types wherever an appropriate type is available. There is no compatibility issue since new parsers for ATF/XML have to be written anyway. These may directly map the ASAM data types to the ASAM ODS data types that are still used in e.g. the OO-API or the RPC-API.

There are some data types used that are not available in the list of ASAM data types; in that case the ASAM ODS data types have been used. They are

- T_DATE: an ASCII string based data type used to specify any date and time information with unlimited time resolution.
- T_EXTERNALREFERENCE: a data type consisting of three strings (a description, a MIME type, and a location (e.g. an URL)); it is used for referencing items that reside outside the ODS server.

The data types used in ATF/XML files are specified in the ASAM data types specification document (which can be requested from the ASAM e.V.) and in the ASAM ODS specification, section 2.5..

All data types used are further mapped to XML standard data types to provide standard XML parsers enough information for validity checking purposes. They are implemented using the native and extended XML schema data types. For many of the data types defined in the base model, there is a corresponding data type in the XML schema (XSD namespace). For several others, a more complex data type can be constructed using the primitives supplied by XSD.

Although though many of the ODS-specific data types have been replaced, they are still referred to by the enumeration data type names.

ASAM Data Types as Defined in ODS base Schema

```
<!--
*****************************************************************
*  Declare base ASAM simple datatypes                           *
*****************************************************************
-->
<xsd:simpleType name="A_ASCIISTRING">
   <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="A_BCD">
   <xsd:restriction base="A_UINT8"/>
</xsd:simpleType>
<xsd:simpleType name="A_BOOLEAN">
   <xsd:restriction base="xsd:boolean"/>
</xsd:simpleType>
<xsd:simpleType name="A_COMPLEX32">
   <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="A_COMPLEX64">
   <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="A_COUNTRY">
   <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="A_FLOAT32">
   <xsd:restriction base="xsd:float"/>
</xsd:simpleType>
```

```
<xsd:simpleType name="A_FLOAT64">
   <xsd:restriction base="xsd:double"/>
</xsd:simpleType>
<xsd:simpleType name="A_INT8">
   <xsd:restriction base="xsd:byte"/>
</xsd:simpleType>
<xsd:simpleType name="A_INT16">
   <xsd:restriction base="xsd:int16"/>
</xsd:simpleType>
<xsd:simpleType name="A_INT32">
   <xsd:restriction base="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="A_INT64">
   <xsd:restriction base="xsd:long"/>
</xsd:simpleType>
<xsd:simpleType name="A_LANGUAGE">
   <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<!–
*****************************************************************
* Note:  Unsigned integer rely on base elements that are signed
*****************************************************************
-->
<xsd:simpleType name="A_UINT8">
   <xsd:restriction base="A_INT8"/>
</xsd:simpleType>
<xsd:simpleType name="A_UINT16">
   <xsd:restriction base="A_INT16"/>
</xsd:simpleType>
<xsd:simpleType name="A_UINT32">
   <xsd:restriction base="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="A_UINT64">
   <xsd:restriction base="A_INT64"/>
</xsd:simpleType>
<!--
*****************************************************************
*     Declare base ASAM complex  datatypes
*****************************************************************
-->
<xsd:complexType name="A_ASCIIFIELD">
   <xsd:sequence>
      <xsd:element name="length" type="A_INT32"/>
      <xsd:element name="sequence" type="A_ASCIISTRING"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="A_BITFIELD">
   <xsd:sequence>
      <xsd:element name="length" type="A_INT16"/>
      <xsd:element name="sequence" type="DS_INT8"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="A_BYTEFIELD">
   <xsd:sequence>
      <xsd:element name="length" type="A_UINT32"/>
      <xsd:element name="sequence" type="DS_INT8"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="A_ENUM">
   <xsd:sequence>
      <xsd:element name="length" type="A_UINT32"/>
   </xsd:sequence>
</xsd:complexType>
```

6.11 ODS BASE SCHEMA FILE

The contents of the ODS base schema file follow. This file may also be downloaded from the ASAM ODS web site.

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="http://www.asam.net"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.asam.net"
elementFormDefault="qualified">
    <xsd:annotation>
        <xsd:documentation>

        ASAM-ODS Base Model Architecture
        W3C Schema Version 1.0
        Copyright 2003 -      Association For Standardization of Automation and Measuring Systems.
                              All rights reserved.

        Revision History

        20 March 2003  - Mark Quinsland  mark.quinsland@highqsoft.com
        Initial Version

        15 April 2003  - Mark Quinsland  mark.quinsland@highqsoft.com
        Adapted to comply with ASAM Data Type Harmonization project.

                Deprecated          New
                T_BYTE              A_INT8
                T_SHORT             A_INT16
                T_LONG          A_INT32
                T_LONGLONG          A_INT64
                T_FLOAT         A_FLOAT32
                T_DOUBLE            A_FLOAT64
                T_COMPLEX           A_COMPLEX32
                T_DCOMPLEX          A_COMPLEX64
                T_BOOLEAN           A_BOOLEAN
                T_STRING            A_ASCIISTRING

         15 June 2003  - Mark Quinsland  mark.quinsland@highqsoft.com
            Added element definitions for instance attributes.
            Added element definitions for application enumerations
            Removed empty definition for instance data

        8 August 2003  - Mark Quinsland  mark.quinsland@highqsoft.com
            Added attribute for schema version.
            Added locale element.
            ODS data types replaced by harmonized ASAM datatypes labeled as deprecated.
            Added t_id and t_ref types to allow xPointer enforcement of relationships between elements.
            Added inverse_name to the relation_attribute complex type.
            t_blob deprecated – replaced by complex type ct_blob


          14 October 2003  - Mark Quinsland  mark.quinsland@highqsoft.com
             Added several data types from ASAM Data Type Harmonization project.

                A_UINT8         8 bit unsigned integer
                A_UINT16        16 bit unsigned integer
                   A_UINT32        32 bit unsigned integer
                A_UINT64      64 bit unsigned integer
                A_ASCIIFIELD   Sequence of up to 2048 bits
                A_BITFIELD   Sequence of up to 2048 bits
                A_BCD          Decimal digits represented by 4 binary digits
                A_ENUM        value used for the description of enumeration values.

              Corrected definition of A_BYTEFIELD to separate length from values

              removed Deprecated ODS Data Types replaced by ASAM data types
              Deprecated                Replaced By
              T_BYTE                A_INT8
              T_SHORT               A_INT16
              T_LONG            A_INT32
              T_LONGLONG        A_INT64
              T_FLOAT           A_FLOAT32
              T_DOUBLE              A_FLOAT64
              T_COMPLEX          A_COMPLEX32
              T_DCOMPLEX         A_COMPLEX64
              T_BOOLEAN          A_BOOLEAN
              T_STRING              A_ASCIISTRING

            7 December 2003  - Mark Quinsland  mark.quinsland@highqsoft.com
             Restored ODS Data Type Enumerations.
             Note:  The datatype enumerations  are not to be confused with the name
             of the datatypes.  They are merely simple names for integers that indicate
             data types.   For example,  the datatype enumeration DT_DOUBLE is merely
             shorthand for the integer 7 which further signifies that the datatype is T_DOUBLE /
A_FLOAT64.
              While T_DOUBLE datatypes may still be used within the ODS API, they may
             not be used in ODS ATF/XML and should be replaced by A_FLOAT64.


  </xsd:documentation>
  </xsd:annotation>
  <!--
  *****************************************************************************************
  *                               declare root element & type                             *
  *****************************************************************************************
  -->
  <xsd:element name="atfx_file">
      <xsd:complexType>
          <xsd:sequence>
              <xsd:element ref="documentation" minOccurs="0"/>
              <xsd:element name="locale" type="A_ASCIISTRING"/>
              <xsd:element name="base_model_version" type="A_ASCIISTRING"/>
              <xsd:element ref="files" minOccurs="0"/>
              <xsd:element ref="application_model"/>
              <xsd:element ref="instance_data" minOccurs="0"/>
              <!—
              reserve area for instance elements which are not validated by this  schema
              -->
          </xsd:sequence>
          <xsd:attribute name="version" type="A_ASCIISTRING" use="required"/>
      </xsd:complexType>
  </xsd:element>
  <!--
  *****************************************************************************************
```

```
*                         declare documentation components                         *
************************************************************************************
-->
<xsd:element name="documentation">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="exported_by" type="A_ASCIISTRING" minOccurs="0"/>
            <xsd:element name="exporter" type="A_ASCIISTRING" minOccurs="0"/>
            <xsd:element name="export_date_time" type="A_ASCIISTRING" minOccurs="0"/>
            <xsd:element name="exporter_version" type="A_ASCIISTRING" minOccurs="0"/>
            <xsd:element name="short_description" type="A_ASCIISTRING" minOccurs="0"/>
            <xsd:element name="long_description" type="A_ASCIISTRING" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<!--
************************************************************************************
*                    declare external file components                              *
************************************************************************************
-->
<xsd:element name="files">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="component" type="filecomponent" nillable="true"
maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<!--
************************************************************************************
*                    declare model meta-data components                            *
************************************************************************************
-->
<xsd:element name="application_model">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="application_element" type="application_element" nillable="true"
maxOccurs="unbounded"/>
            <xsd:element name="application_enumeration" type="application_enumeration"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<!--
************************************************************************************
*             Define base structure for application element definitions            *
************************************************************************************
-->
<xsd:complexType name="application_element">
    <xsd:sequence>
        <xsd:element name="name" type="A_ASCIISTRING"/>
        <xsd:element name="basetype" type="elemtype_enum"/>
        <xsd:element name="application_attribute" type="application_attribute" minOccurs="0"
maxOccurs="unbounded"/>
        <xsd:element name="relation_attribute" type="relation_attribute" minOccurs="0"
maxOccurs="unbounded"/>
    </xsd:sequence>
```

```
</xsd:complexType>
<!--
*******************************************************************************************
*           Define base structure for application attribute definitions                 *
*******************************************************************************************
-->
<xsd:complexType name="application_attribute">
    <xsd:sequence>
        <xsd:element name="name" type="A_ASCIISTRING"/>
        <xsd:element name="base_attribute" type="base_attributetype_enum" minOccurs="0"/>
        <xsd:element name="datatype" type="datatype_enum" minOccurs="0"/>
        <xsd:element name="enumeration_type" type="A_ASCIISTRING" minOccurs="0"/>
        <xsd:element name="autogenerate" type="boolean_enum" minOccurs="0"/>
        <xsd:element name="obligatory" type="boolean_enum" minOccurs="0"/>
        <xsd:element name="unique" type="boolean_enum" minOccurs="0"/>
        <xsd:element name="length" type="A_INT32" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
<!--
*******************************************************************************************
*           Define base structure for application relation definitions                  *
*******************************************************************************************
-->
<!--  Define base Application Relation Attributes -->
<xsd:complexType name="relation_attribute">
    <xsd:sequence>
        <xsd:element name="name" type="A_ASCIISTRING"/>
        <xsd:element name="ref_to" type="A_ASCIISTRING"/>
        <xsd:element name="base_relation" type="base_relation_enum" minOccurs="0"/>
        <xsd:element name="min_occurs" type="A_INT32" minOccurs="0"/>
        <xsd:element name="max_occurs" type="A_ASCIISTRING" minOccurs="0"/>
        <xsd:element name="inverse_name" type="A_ASCIISTRING" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
<!--
*******************************************************************************************
*   User-Defined Application Enumeration Objects
*******************************************************************************************
 -->
<xsd:complexType name="application_enumeration">
    <xsd:sequence>
        <xsd:element name="name" type="A_ASCIISTRING"/>
        <xsd:element name="item" type="enumeration_item" nillable="true"
maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="enumeration_item">
    <xsd:sequence>
        <xsd:element name="name" type="A_ASCIISTRING"/>
        <xsd:element name="value" type="A_ASCIISTRING"/>
    </xsd:sequence>
</xsd:complexType>
<!--
*******************************************************************************************
*                   declare instance data components                                    *
*******************************************************************************************
-->
```

```
<xsd:element name="instance_data">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:any processContents="skip" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<!--
*****************************************************************************************
*       Declare base ASAM simple  datatypes                                             *
*****************************************************************************************
-->
<xsd:simpleType name="A_ASCIISTRING">
    <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
<xsd:simpleType name="A_BCD">
    <xsd:restriction base="A_UINT8"/>
</xsd:simpleType>
<xsd:simpleType name="A_BOOLEAN">
    <xsd:restriction base="xsd:boolean"/>
</xsd:simpleType>
<xsd:simpleType name="A_COMPLEX32">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="A_COMPLEX64">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="A_COUNTRY">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="A_FLOAT32">
    <xsd:restriction base="xsd:float"/>
</xsd:simpleType>
<xsd:simpleType name="A_FLOAT64">
    <xsd:restriction base="xsd:double"/>
</xsd:simpleType>
<xsd:simpleType name="A_INT8">
    <xsd:restriction base="xsd:byte"/>
</xsd:simpleType>
<xsd:simpleType name="A_INT16">
    <xsd:restriction base="xsd:integer"/>
</xsd:simpleType>
<xsd:simpleType name="A_INT32">
    <xsd:restriction base="xsd:integer"/>
</xsd:simpleType>
<xsd:simpleType name="A_INT64">
    <xsd:restriction base="xsd:long"/>
</xsd:simpleType>
<xsd:simpleType name="A_LANGUAGE">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<!--
*****************************************************************************************
*       Note:  Unsigned integer rely on base elements that are signed                   *
*****************************************************************************************
-->
<xsd:simpleType name="A_UINT8">
```

```
    <xsd:restriction base="xsd:byte"/>
</xsd:simpleType>
<xsd:simpleType name="A_UINT16">
    <xsd:restriction base="xsd:short"/>
</xsd:simpleType>
<xsd:simpleType name="A_UINT32">
    <xsd:restriction base="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="A_UINT64">
    <xsd:restriction base="A_INT64"/>
</xsd:simpleType>
<!--
*********************************************************************************************
*        Declare base ASAM complex  datatypes                                               *
*********************************************************************************************
-->
<xsd:complexType name="A_ASCIIFIELD">
    <xsd:sequence>
        <xsd:element name="length" type="A_INT32"/>
        <xsd:element name="sequence" type="A_ASCIISTRING"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="A_BITFIELD">
    <xsd:sequence>
        <xsd:element name="length" type="A_INT16"/>
        <xsd:element name="sequence" type="DS_INT8"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="A_BYTEFIELD">
    <xsd:sequence>
        <xsd:element name="length" type="A_UINT32"/>
        <xsd:element name="sequence" type="DS_INT8"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="A_ENUM">
    <xsd:sequence>
        <xsd:element name="length" type="A_UINT32"/>
    </xsd:sequence>
</xsd:complexType>
<!--
*********************************************************************************************
*        Additional building blocks for instance element definitions                        *
*********************************************************************************************
-->
<xsd:simpleType name="t_character">
    <xsd:restriction base="A_ASCIISTRING">
        <xsd:length value="1"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="t_date">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="t_id">
    <xsd:restriction base="xsd:ID"/>
</xsd:simpleType>
<xsd:simpleType name="t_ref">
    <xsd:restriction base="xsd:anyURI"/>
```

```
</xsd:simpleType>
<xsd:simpleType name="t_ref_list">
    <xsd:list itemType="t_ref"/>
</xsd:simpleType>
<!--
****************************************************************************************************
*   The next level of building block
****************************************************************************************************
 -->
<xsd:simpleType name="DT_DATE">
    <xsd:restriction base="t_date"/>
</xsd:simpleType>
<xsd:simpleType name="posLongInt">
    <xsd:restriction base="xsd:unsignedLong">
        <xsd:minExclusive value="0"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="values_float64">
    <xsd:list itemType="A_FLOAT64"/>
</xsd:simpleType>
<xsd:simpleType name="values_float32">
    <xsd:list itemType="A_FLOAT32"/>
</xsd:simpleType>
<xsd:simpleType name="values_date">
    <xsd:list itemType="t_date"/>
</xsd:simpleType>
<xsd:simpleType name="values_int32">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="values_int64">
    <xsd:list itemType="A_INT64"/>
</xsd:simpleType>
<xsd:simpleType name="values_int16">
    <xsd:list itemType="A_INT16"/>
</xsd:simpleType>
<xsd:simpleType name="values_character">
    <xsd:list itemType="t_character"/>
</xsd:simpleType>
<!--
****************************************************************************************************
*  Subtypes of instance_attribute
****************************************************************************************************
 -->
<xsd:simpleType name="asciistring_attribute">
    <xsd:list itemType="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="float32_attribute">
    <xsd:list itemType="A_FLOAT32"/>
</xsd:simpleType>
<xsd:simpleType name="float64_attribute">
    <xsd:list itemType="A_FLOAT64"/>
</xsd:simpleType>
<xsd:simpleType name="int8_attribute">
    <xsd:list itemType="A_INT8"/>
</xsd:simpleType>
<xsd:simpleType name="int16_attribute">
    <xsd:list itemType="A_INT16"/>
```

```
</xsd:simpleType>
<xsd:simpleType name="int32_attribute">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="time_attribute">
    <xsd:list itemType="t_date"/>
</xsd:simpleType>
<!--
*************************************************************************************************
* Sequence Subtypes of instance_attribute
*************************************************************************************************
 -->
<xsd:simpleType name="DS_ASCIISTRING">
    <xsd:list itemType="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="DS_BLOB">
    <xsd:list itemType="A_INT8"/>
</xsd:simpleType>
<xsd:simpleType name="DS_BOOLEAN">
    <xsd:list itemType="A_BOOLEAN"/>
</xsd:simpleType>
<xsd:simpleType name="DS_BYTESTR">
    <xsd:list itemType="A_INT8"/>
</xsd:simpleType>
<xsd:simpleType name="DS_COMPLEX32">
    <xsd:list itemType="A_COMPLEX32"/>
</xsd:simpleType>
<xsd:simpleType name="DS_COMPLEX64">
    <xsd:list itemType="A_COMPLEX64"/>
</xsd:simpleType>
<xsd:simpleType name="DS_DATE">
    <xsd:list itemType="t_date"/>
</xsd:simpleType>
<xsd:simpleType name="DS_DIMENSION">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="DS_FLOAT32">
    <xsd:list itemType="A_FLOAT32"/>
</xsd:simpleType>
<xsd:simpleType name="DS_FLOAT64">
    <xsd:list itemType="A_FLOAT64"/>
</xsd:simpleType>
<xsd:simpleType name="DS_ID">
    <xsd:list itemType="A_INT8"/>
</xsd:simpleType>
<xsd:simpleType name="DS_INT8">
    <xsd:list itemType="A_INT8"/>
</xsd:simpleType>
<xsd:simpleType name="DS_INT16">
    <xsd:list itemType="A_INT16"/>
</xsd:simpleType>
<xsd:simpleType name="DS_INT32">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="DS_INT64">
    <xsd:list itemType="A_INT64"/>
</xsd:simpleType>
```

```
<!--
************************************************************************************************
*  Subtypes of instance_attribute
************************************************************************************************
 -->
<!—Simple Base Attribute Types  -->
<xsd:simpleType name="asam_path">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="attribute_name">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="attribute_type">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="base_element_name">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="entity_name">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="integerSequence">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="max_test_level">
    <xsd:restriction base="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="objecttype">
    <xsd:restriction base="A_INT64"/>
</xsd:simpleType>
<xsd:simpleType name="rule_text">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="time_sequence">
    <xsd:list itemType="t_date"/>
</xsd:simpleType>
<!--
************************************************************************************************
*  Complex types for instance attributes
************************************************************************************************
 -->
<xsd:complexType name="instance_attributes">
    <xsd:choice maxOccurs="unbounded">
        <xsd:element name="inst_attr_asciistring" type="inst_attr_asciistring"/>
        <xsd:element name="inst_attr_asciistring_seq" type="inst_attr_asciistring_seq"/>
        <xsd:element name="inst_attr_complex32" type="inst_attr_complex32"/>
        <xsd:element name="inst_attr_complex32_seq" type="inst_attr_complex32_seq"/>
        <xsd:element name="inst_attr_complex64" type="inst_attr_complex64"/>
        <xsd:element name="inst_attr_complex64_seq" type="inst_attr_complex64_seq"/>
        <xsd:element name="inst_attr_float32" type="inst_attr_float32"/>
        <xsd:element name="inst_attr_float32_seq" type="inst_attr_float32_seq"/>
        <xsd:element name="inst_attr_float64" type="inst_attr_float64"/>
        <xsd:element name="inst_attr_float64_seq" type="inst_attr_float64_seq"/>
        <xsd:element name="inst_attr_int8" type="inst_attr_int8"/>
        <xsd:element name="inst_attr_int8_seq" type="inst_attr_int8_seq"/>
        <xsd:element name="inst_attr_int16" type="inst_attr_int16"/>
        <xsd:element name="inst_attr_int16_seq" type="inst_attr_int16_seq"/>
```

```
            <xsd:element name="inst_attr_int32" type="inst_attr_int32"/>
            <xsd:element name="inst_attr_int32_seq" type="inst_attr_int32_seq"/>
            <xsd:element name="inst_attr_time_seq" type="inst_attr_time_seq"/>
            <xsd:element name="inst_attr_time_seq" type="inst_attr_time_seq"/>
        </xsd:choice>
    </xsd:complexType>
    <xsd:complexType name="inst_attr_asciistring">
        <xsd:simpleContent>
            <xsd:extension base="A_ASCIISTRING">
                <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="inst_attr_asciistring_seq">
        <xsd:complexContent>
            <xsd:extension base="string_sequence">
                <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="inst_attr_complex32">
        <xsd:simpleContent>
            <xsd:extension base="A_COMPLEX32">
                <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="inst_attr_complex32_seq">
        <xsd:simpleContent>
            <xsd:extension base="A_COMPLEX32">
                <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="inst_attr_complex64">
        <xsd:simpleContent>
            <xsd:extension base="A_COMPLEX64">
                <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="inst_attr_complex64_seq">
        <xsd:simpleContent>
            <xsd:extension base="DS_COMPLEX64">
                <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="inst_attr_float32">
        <xsd:simpleContent>
            <xsd:extension base="A_FLOAT32">
                <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="inst_attr_float32_seq">
        <xsd:simpleContent>
        <xsd:extension base="DS_FLOAT32">
            <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="inst_attr_float64">
    <xsd:simpleContent>
        <xsd:extension base="A_FLOAT64">
            <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="inst_attr_float64_seq">
    <xsd:simpleContent>
        <xsd:extension base="DS_FLOAT64">
            <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="inst_attr_int8">
    <xsd:simpleContent>
        <xsd:extension base="A_INT8">
            <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="inst_attr_int8_seq">
    <xsd:simpleContent>
        <xsd:extension base="DS_INT8">
            <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="inst_attr_int16">
    <xsd:simpleContent>
        <xsd:extension base="A_INT16">
            <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="inst_attr_int16_seq">
    <xsd:simpleContent>
        <xsd:extension base="DS_INT16">
            <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="inst_attr_int32">
    <xsd:simpleContent>
        <xsd:extension base="A_INT32">
            <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="inst_attr_int32_seq">
    <xsd:simpleContent>
        <xsd:extension base="DS_INT32">
```

```
                <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="inst_attr_int64">
        <xsd:simpleContent>
            <xsd:extension base="A_INT64">
                <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="inst_attr_int64_seq">
        <xsd:simpleContent>
            <xsd:extension base="DS_INT64">
                <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="inst_attr_time">
        <xsd:simpleContent>
            <xsd:extension base="DT_DATE">
                <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="inst_attr_time_seq">
        <xsd:simpleContent>
            <xsd:extension base="DS_DATE">
                <xsd:attribute name="name" type="A_ASCIISTRING" use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <!--
******************************************************************************************
    *  Complex types for elements
******************************************************************************************
     -->
    <xsd:complexType name="complex32_attribute">
        <xsd:sequence>
            <xsd:element name="realPart" type="A_FLOAT32"/>
            <xsd:element name="imaginaryPart" type="A_FLOAT32"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="component">
        <xsd:sequence>
            <xsd:element name="identifier" type="A_ASCIISTRING"/>
            <xsd:element name="datatype" type="component_datatype"/>
            <xsd:element name="length" type="A_INT32"/>
            <xsd:element name="description" type="A_ASCIISTRING" minOccurs="0"/>
            <xsd:element name="inioffset" type="A_INT32" minOccurs="0"/>
            <xsd:element name="blocksize" type="A_INT32" minOccurs="0"/>
            <xsd:element name="valperblock" type="A_INT32" minOccurs="0"/>
            <xsd:element name="valoffsets" type="A_INT32" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="ct_blob">
        <xsd:sequence>
```

```
            <xsd:element name="text" type="A_ASCIISTRING"/>
            <xsd:element name="bytefield" type="A_BYTEFIELD"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="complex64_attribute">
        <xsd:sequence>
            <xsd:element name="realPart" type="A_FLOAT64"/>
            <xsd:element name="imaginaryPart" type="A_FLOAT64"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="filecomponent">
        <xsd:sequence>
            <xsd:element name="identifier" type="A_ASCIISTRING"/>
            <xsd:element name="filename" type="A_ASCIISTRING"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="real_and_unit">
        <xsd:sequence>
            <xsd:element name="values" type="values"/>
            <xsd:element name="unit" type="AoUnit"/>
            <xsd:element name="quantity" type="AoUnit" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="string_sequence">
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="s" type="A_ASCIISTRING"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="t_externalreference">
        <xsd:sequence>
            <xsd:element name="description" type="A_ASCIISTRING"/>
            <xsd:element name="mimetype" type="A_ASCIISTRING"/>
            <xsd:element name="location" type="A_ASCIISTRING"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="values">
        <xsd:choice>
            <xsd:element name="values" type="values_float64"/>
        </xsd:choice>
    </xsd:complexType>
    <xsd:complexType name="value_with_unit">
        <xsd:sequence>
            <xsd:element name="values" type="values"/>
            <xsd:element name="unit" type="AoUnit"/>
            <xsd:element name="quantity" type="AoUnit" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
    <!--
    **************************************************************************************************
    *  Base Attributes
    **************************************************************************************************
     -->
    <xsd:simpleType name="appl_element_name">
        <xsd:restriction base="A_ASCIISTRING"/>
    </xsd:simpleType>
    <xsd:simpleType name="average">
        <xsd:list itemType="A_FLOAT64"/>
```

```
</xsd:simpleType>
<xsd:simpleType name="blocksize">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="component_datatype">
    <xsd:list itemType="component_datatype_enum"/>
</xsd:simpleType>
<xsd:simpleType name="current_exp">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="current_exp_den">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="default_datatype">
    <xsd:list itemType="datatype_enum"/>
</xsd:simpleType>
<xsd:simpleType name="default_dimension">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="default_mq_name">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="default_rank">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="default_type_size">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="description">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:complexType name="external_reference">
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="externalreference" type="t_externalreference"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="factor">
    <xsd:list itemType="A_FLOAT64"/>
</xsd:simpleType>
<xsd:simpleType name="filename_url">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="flags">
    <xsd:list itemType="A_INT16"/>
</xsd:simpleType>
<xsd:simpleType name="flags_filename_url">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="flags_start_offset">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="generation_parameters">
    <xsd:list itemType="A_FLOAT64"/>
</xsd:simpleType>
<xsd:simpleType name="global_flag">
    <xsd:list itemType="A_INT16"/>
</xsd:simpleType>
```

```
<xsd:simpleType name="id">
    <xsd:list itemType="A_INT64"/>
</xsd:simpleType>
<xsd:simpleType name="independent">
    <xsd:list itemType="A_BOOLEAN"/>
</xsd:simpleType>
<xsd:simpleType name="instance_id">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="interpolation">
    <xsd:list itemType="interpolation_enum"/>
</xsd:simpleType>
<xsd:simpleType name="length_exp">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="length_exp_den">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="level">
    <xsd:list itemType="A_INT16"/>
</xsd:simpleType>
<xsd:simpleType name="luminous_intensity_exp">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="luminous_intensity_exp_den">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="mass_exp">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="mass_exp_den">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="maximum">
    <xsd:list itemType="A_FLOAT64"/>
</xsd:simpleType>
<xsd:simpleType name="measurement_begin">
    <xsd:list itemType="t_date"/>
</xsd:simpleType>
<xsd:simpleType name="measurement_end">
    <xsd:list itemType="t_date"/>
</xsd:simpleType>
<xsd:simpleType name="mime_type">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="minimum">
    <xsd:list itemType="A_FLOAT64"/>
</xsd:simpleType>
<xsd:simpleType name="molar_amount_exp">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="molar_amount_exp_den">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="name">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
```

```
<xsd:simpleType name="number_of_rows">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="offset">
    <xsd:list itemType="A_FLOAT64"/>
</xsd:simpleType>
<xsd:simpleType name="parameter_datatype">
    <xsd:list itemType="parameter_datatype_enum"/>
</xsd:simpleType>
<xsd:simpleType name="password">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="pvalue">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="rank">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="sequence_representation">
    <xsd:restriction base="seq_rep_enum"/>
</xsd:simpleType>
<xsd:simpleType name="standard_deviation">
    <xsd:list itemType="A_FLOAT64"/>
</xsd:simpleType>
<xsd:simpleType name="start_offset">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="temperature_exp">
    <xsd:restriction base="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="temperature_exp_den">
    <xsd:restriction base="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="time_exp">
    <xsd:restriction base="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="time_exp_den">
    <xsd:restriction base="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="type_size">
    <xsd:restriction base="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="value_offsets">
    <xsd:list itemType="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="superuser_flag">
    <xsd:restriction base="A_BOOLEAN"/>
</xsd:simpleType>
<xsd:simpleType name="ref_appl_elem_name">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="rights">
    <xsd:restriction base="A_INT16"/>
</xsd:simpleType>
<xsd:simpleType name="version">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
```

```
<xsd:simpleType name="version_date">
    <xsd:restriction base="t_date"/>
</xsd:simpleType>
<xsd:simpleType name="value">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="valuesperblock">
    <xsd:restriction base="A_INT32"/>
</xsd:simpleType>
<xsd:simpleType name="quantity_class_enum">
    <xsd:restriction base="A_ASCIISTRING">
        <xsd:enumeration value="measured"/>
        <xsd:enumeration value="state"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
******************************************************************************************
*  Enumerations
******************************************************************************************
 -->
<xsd:simpleType name="base_attributetype_enum">
    <xsd:restriction base="A_ASCIISTRING">
        <xsd:enumeration value="appl_element_id"/>
        <xsd:enumeration value="attribute_name"/>
        <xsd:enumeration value="average"/>
        <xsd:enumeration value="block_size"/>
        <xsd:enumeration value="component_datatype"/>
        <xsd:enumeration value="current_exp"/>
        <xsd:enumeration value="current_exp_den"/>
        <xsd:enumeration value="default_datatype"/>
        <xsd:enumeration value="datatype_enum"/>
        <xsd:enumeration value="default_dimension"/>
        <xsd:enumeration value="default_mq_name"/>
        <xsd:enumeration value="default_rank"/>
        <xsd:enumeration value="default_type_size"/>
        <xsd:enumeration value="description"/>
        <xsd:enumeration value="dimension"/>
        <xsd:enumeration value="external_reference"/>
        <xsd:enumeration value="factor"/>
        <xsd:enumeration value="filename_url"/>
        <xsd:enumeration value="flags"/>
        <xsd:enumeration value="flags_filename_url"/>
        <xsd:enumeration value="flags_start_offset"/>
        <xsd:enumeration value="generation_parameters"/>
        <xsd:enumeration value="global_flag"/>
        <xsd:enumeration value="id"/>
        <xsd:enumeration value="independent"/>
        <xsd:enumeration value="instance_id"/>
        <xsd:enumeration value="interpolation"/>
        <xsd:enumeration value="length_exp"/>
        <xsd:enumeration value="length_exp_den"/>
        <xsd:enumeration value="length_iin_bytes"/>
        <xsd:enumeration value="level"/>
        <xsd:enumeration value="luminous_intensity_exp"/>
        <xsd:enumeration value="luminous_intensity_exp_den"/>
        <xsd:enumeration value="mass_exp"/>
        <xsd:enumeration value="mass_exp_den"/>
```

```
            <xsd:enumeration value="maximum"/>
            <xsd:enumeration value="measurement_begin"/>
            <xsd:enumeration value="measurement_end"/>
            <xsd:enumeration value="mime_type"/>
            <xsd:enumeration value="minimum"/>
            <xsd:enumeration value="molar_amount_exp"/>
            <xsd:enumeration value="molar_amount_exp_den"/>
            <xsd:enumeration value="name"/>
            <xsd:enumeration value="number_of_rows"/>
            <xsd:enumeration value="ordinal_number"/>
            <xsd:enumeration value="offset"/>
            <xsd:enumeration value="parameter_datatype"/>
            <xsd:enumeration value="pvalue"/>
            <xsd:enumeration value="password"/>
            <xsd:enumeration value="rank"/>
            <xsd:enumeration value="ref_appl_elem_name"/>
            <xsd:enumeration value="rights"/>
            <xsd:enumeration value="sequence_representation"/>
            <xsd:enumeration value="standard_deviation"/>
            <xsd:enumeration value="start_offset"/>
            <xsd:enumeration value="superuser_flag"/>
            <xsd:enumeration value="temperature_exp"/>
            <xsd:enumeration value="temperature_exp_den"/>
            <xsd:enumeration value="time_exp"/>
            <xsd:enumeration value="time_exp_den"/>
            <xsd:enumeration value="type_size"/>
            <xsd:enumeration value="typespec_enum"/>
            <xsd:enumeration value="value_offsets"/>
            <xsd:enumeration value="version"/>
            <xsd:enumeration value="version_date"/>
            <xsd:enumeration value="value"/>
            <xsd:enumeration value="value_offset"/>
            <xsd:enumeration value="value_type"/>
            <xsd:enumeration value="values_float64"/>
            <xsd:enumeration value="valuesperblock"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="base_relation_enum">
        <xsd:restriction base="A_ASCIISTRING">
            <xsd:enumeration value="true"/>
            <xsd:enumeration value="false"/>
            <xsd:enumeration value="alias_names"/>
            <xsd:enumeration value="attribute_mapping"/>
            <xsd:enumeration value="channel"/>
            <xsd:enumeration value="children"/>
            <xsd:enumeration value="datatype"/>
            <xsd:enumeration value="default_unit"/>
            <xsd:enumeration value="entity_mapping"/>
            <xsd:enumeration value="equipments"/>
            <xsd:enumeration value="groups"/>
            <xsd:enumeration value="is_scaled_by"/>
            <xsd:enumeration value="local_columns"/>
            <xsd:enumeration value="meaning_of_aliases"/>
            <xsd:enumeration value="measurement"/>
            <xsd:enumeration value="measurement_quantities"/>
            <xsd:enumeration value="measurement_quantity"/>
            <xsd:enumeration value="parent_sequence"/>
            <xsd:enumeration value="parent_test"/>
            <xsd:enumeration value="parent_unit_under_test"/>
            <xsd:enumeration value="predecessor"/>
            <xsd:enumeration value="phys_dimension"/>
            <xsd:enumeration value="quantity"/>
            <xsd:enumeration value="quantities"/>
            <xsd:enumeration value="quantity_instance"/>
            <xsd:enumeration value="sequences"/>
            <xsd:enumeration value="submatrices"/>
            <xsd:enumeration value="submatrix"/>
            <xsd:enumeration value="successors"/>
            <xsd:enumeration value="test"/>
            <xsd:enumeration value="tests"/>
            <xsd:enumeration value="unit"/>
            <xsd:enumeration value="units"/>
            <xsd:enumeration value="unit_instance"/>
            <xsd:enumeration value="units_under_test"/>
            <xsd:enumeration value="users"/>
            <xsd:enumeration value="uuts"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="elemtype_enum">
        <xsd:restriction base="A_ASCIISTRING">
            <xsd:enumeration value="ACL"/>
            <xsd:enumeration value="ACLA"/>
            <xsd:enumeration value="ACLI"/>
            <xsd:enumeration value="ACLTemplate"/>
            <xsd:enumeration value="AoAttributeMap"/>
            <xsd:enumeration value="AoEnvironment"/>
            <xsd:enumeration value="AoExternalComponent"/>
            <xsd:enumeration value="AoLocalColumn"/>
            <xsd:enumeration value="AoLog"/>
            <xsd:enumeration value="AoMeasurement"/>
            <xsd:enumeration value="AoMeasurementQuantity"/>
            <xsd:enumeration value="AoNameMap"/>
            <xsd:enumeration value="AoParameter"/>
            <xsd:enumeration value="AoParameterSet"/>
            <xsd:enumeration value="AoPhysicalDimension"/>
            <xsd:enumeration value="AoQuantity"/>
            <xsd:enumeration value="AoQuantityGroup"/>
            <xsd:enumeration value="AoSubmatrix"/>
            <xsd:enumeration value="AoSubTest"/>
            <xsd:enumeration value="AoTest"/>
            <xsd:enumeration value="AoTestEquipment"/>
            <xsd:enumeration value="AoTestEquipmentPart"/>
            <xsd:enumeration value="AoTestSequence"/>
            <xsd:enumeration value="AoTestSequencePart"/>
            <xsd:enumeration value="AoUnitUnderTest"/>
            <xsd:enumeration value="AoUnitUnderTestPart"/>
            <xsd:enumeration value="AoUnit"/>
            <xsd:enumeration value="AoUnitGroup"/>
            <xsd:enumeration value="AoUser"/>
            <xsd:enumeration value="AoUserGroup"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="boolean_enum">
        <xsd:restriction base="A_ASCIISTRING">
```

```
            <xsd:enumeration value="true"/>
            <xsd:enumeration value="false"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="component_datatype_enum">
        <xsd:restriction base="A_ASCIISTRING">
            <xsd:enumeration value="A_ASCIISTRING"/>
            <xsd:enumeration value="A_BITFIELD"/>
            <xsd:enumeration value="A_BOOLEAN"/>
            <xsd:enumeration value="A_BYTEFIELD"/>
            <xsd:enumeration value="A_FLOAT32"/>
            <xsd:enumeration value="A_FLOAT32_BEO"/>
            <xsd:enumeration value="A_FLOAT64"/>
            <xsd:enumeration value="A_FLOAT64_BEO"/>
            <xsd:enumeration value="A_INT8"/>
            <xsd:enumeration value="A_INT16"/>
            <xsd:enumeration value="A_INT16_BEO"/>
            <xsd:enumeration value="A_INT32"/>
            <xsd:enumeration value="A_INT32_BEO"/>
            <xsd:enumeration value="A_INT64"/>
            <xsd:enumeration value="A_INT64_BEO"/>
            <xsd:enumeration value="A_UINT16"/>
            <xsd:enumeration value="A_UINT32"/>
            <xsd:enumeration value="A_UINT8"/>
            <xsd:enumeration value="DT_ENUM"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="datatype_enum">
        <xsd:restriction base="A_ASCIISTRING">
            <xsd:enumeration value="DS_BOOLEAN"/>
            <xsd:enumeration value="DS_BYTE"/>
            <xsd:enumeration value="DS_BYTESTR"/>
            <xsd:enumeration value="DS_COMPLEX"/>
            <xsd:enumeration value="DS_DATE"/>
            <xsd:enumeration value="DS_DCOMPLEX"/>
            <xsd:enumeration value="DS_DOUBLE"/>
            <xsd:enumeration value="DT_ENUM"/>
            <xsd:enumeration value="DS_EXTERNALREFERENCE"/>
            <xsd:enumeration value="DS_FLOAT"/>
            <xsd:enumeration value="DS_ID"/>
            <xsd:enumeration value="DS_LONG"/>
            <xsd:enumeration value="DS_LONGLONG"/>
            <xsd:enumeration value="DS_SHORT"/>
            <xsd:enumeration value="DS_STRING"/>
            <xsd:enumeration value="DT_BLOB"/>
            <xsd:enumeration value="DT_BOOLEAN"/>
            <xsd:enumeration value="DT_BYTE"/>
            <xsd:enumeration value="DT_BYTESTR"/>
            <xsd:enumeration value="DT_COMPLEX"/>
            <xsd:enumeration value="DT_DATE"/>
            <xsd:enumeration value="DT_DCOMPLEX"/>
            <xsd:enumeration value="DT_DOUBLE"/>
            <xsd:enumeration value="DT_ENUM"/>
            <xsd:enumeration value="DT_EXTERNALREFERENCE"/>
            <xsd:enumeration value="DT_FLOAT"/>
            <xsd:enumeration value="DT_ID"/>
            <xsd:enumeration value="DT_LONG"/>
        <xsd:enumeration value="DT_LONGLONG"/>
        <xsd:enumeration value="DT_SHORT"/>
        <xsd:enumeration value="DT_STRING"/>
        <xsd:enumeration value="DT_UNKNOWN"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="interpolation_enum">
    <xsd:restriction base="A_ASCIISTRING">
        <xsd:enumeration value="no_interpolation"/>
        <xsd:enumeration value="linear_interpolation"/>
        <xsd:enumeration value="application_specific"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="parameter_datatype_enum">
    <xsd:restriction base="A_ASCIISTRING"/>
</xsd:simpleType>
<xsd:simpleType name="seq_rep_enum">
    <xsd:restriction base="A_ASCIISTRING">
        <xsd:enumeration value="explicit"/>
        <xsd:enumeration value="implicit_constant"/>
        <xsd:enumeration value="implicit_linear"/>
        <xsd:enumeration value="implicit_saw"/>
        <xsd:enumeration value="raw_linear"/>
        <xsd:enumeration value="raw_linear_external"/>
        <xsd:enumeration value="raw_linear_calibrated"/>
        <xsd:enumeration value="raw_linear_calibrated_external"/>
        <xsd:enumeration value="raw_polynomial"/>
        <xsd:enumeration value="raw_polynomial_external"/>
        <xsd:enumeration value="formula"/>
        <xsd:enumeration value="external_component"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
**************************************************************************************************
* Abstract declarations for ASAM Base Relations
**************************************************************************************************
 -->
<xsd:complexType name="alias_names" abstract="true"/>
<xsd:complexType name="attribute_mapping" abstract="true"/>
<xsd:complexType name="channel" abstract="true"/>
<xsd:complexType name="children" abstract="true"/>
<xsd:complexType name="datatype" abstract="true"/>
<xsd:complexType name="default_unit" abstract="true"/>
<xsd:complexType name="entity_mapping" abstract="true"/>
<xsd:complexType name="equipments" abstract="true"/>
<xsd:complexType name="groups" abstract="true"/>
<xsd:complexType name="is_scaled_by" abstract="true"/>
<xsd:complexType name="local_columns" abstract="true"/>
<xsd:complexType name="meaning_of_aliases" abstract="true"/>
<xsd:complexType name="measurement" abstract="true"/>
<xsd:complexType name="measurement_quantities" abstract="true"/>
<xsd:complexType name="measurement_quantity" abstract="true"/>
<xsd:complexType name="parameter_set" abstract="true"/>
<xsd:complexType name="parameters" abstract="true"/>
<xsd:complexType name="parent_sequence" abstract="true"/>
<xsd:complexType name="parent_test" abstract="true"/>
<xsd:complexType name="parent_unit_under_test" abstract="true"/>
```

```
    <xsd:complexType name="phys_dimension" abstract="true"/>
    <xsd:complexType name="predecessor" abstract="true"/>
    <xsd:complexType name="quantity_instance" abstract="true"/>
    <xsd:complexType name="sequences" abstract="true"/>
    <xsd:complexType name="submatrices" abstract="true"/>
    <xsd:complexType name="submatrx" abstract="true"/>
    <xsd:complexType name="test" abstract="true"/>
    <xsd:complexType name="tests" abstract="true"/>
    <xsd:complexType name="unit" abstract="true"/>
    <xsd:complexType name="unit_instance" abstract="true"/>
    <xsd:complexType name="units_under_test" abstract="true"/>
    <xsd:complexType name="users" abstract="true"/>
    <xsd:complexType name="uuts" abstract="true"/>
    <!--
    ***********************************************************************************************
    *  Abstract declarations for ASAM Base Elements
    ***********************************************************************************************
     -->
    <xsd:complexType name="AoAny" abstract="true"/>
    <xsd:complexType name="AoAttributeMap" abstract="true"/>
    <xsd:complexType name="AoEnvironment" abstract="true"/>
    <xsd:complexType name="AoExternalComponent" abstract="true"/>
    <xsd:complexType name="AoLocalColumn" abstract="true"/>
    <xsd:complexType name="AoLog" abstract="true"/>
    <xsd:complexType name="AoMeasurement" abstract="true"/>
    <xsd:complexType name="AoMeasurementQuantity" abstract="true"/>
    <xsd:complexType name="AoNameMap" abstract="true"/>
    <xsd:complexType name="AoParameter" abstract="true"/>
    <xsd:complexType name="AoParameterSet" abstract="true"/>
    <xsd:complexType name="AoPhysicalDimension" abstract="true"/>
    <xsd:complexType name="AoQuantity" abstract="true"/>
    <xsd:complexType name="AoQuantityGroup" abstract="true"/>
    <xsd:complexType name="AoSubmatrix" abstract="true"/>
    <xsd:complexType name="AoSubTest" abstract="true"/>
    <xsd:complexType name="AoTest" abstract="true"/>
    <xsd:complexType name="AoTestEquipment" abstract="true"/>
    <xsd:complexType name="AoTestEquipmentPart" abstract="true"/>
    <xsd:complexType name="AoTestSequence" abstract="true"/>
    <xsd:complexType name="AoTestSequencePart" abstract="true"/>
    <xsd:complexType name="AoUnitUnderTest" abstract="true"/>
    <xsd:complexType name="AoUnitUnderTestPart" abstract="true"/>
    <xsd:complexType name="AoUnit" abstract="true"/>
    <xsd:complexType name="AoUnitGroup" abstract="true"/>
    <xsd:complexType name="AoUser" abstract="true"/>
    <xsd:complexType name="AoUserGroup" abstract="true"/>
    <!--
    ***********************************************************************************************
    *  Abstract declarations for Security – Not ASAM Base Elements
    ***********************************************************************************************
     -->
    <xsd:complexType name="ACL" abstract="true"/>
    <xsd:complexType name="ACLA" abstract="true"/>
    <xsd:complexType name="ACLI" abstract="true"/>
    <xsd:complexType name="ACLTemplate" abstract="true"/>
</xsd:schema>
```

6.12 EXAMPLE ATF/XML FILE

The contents of a demo XML file follow. This file is essentially an ATF/XML version of the same demo ATF file that is included with the ATF documentation. This file may be downloaded from the ASAM ODS web site.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
   *****************************************************************************************************
   *                    Example of ODS ATF in XML for Base Model 27                                   *
   *****************************************************************************************************

      ASAM-ODS ATF/XML Example
         Version 1.0
      Copyright 2003 -  Association For Standardization of Automation and Measuring Systems.
          All rights reserved.
      Revision History

      20 March 2003  - Mark Quinsland  mark.quinsland@highqsoft.com
      Initial Version

      22 June 2003  - Mark Quinsland  mark.quinsland@highqsoft.com
      Utilizes new style for defining String sequences.  Utilizes new Instance Attributes.

         22 June 2003  - Mark Quinsland  mark.quinsland@highqsoft.com
      Utilizes new style for defining String sequences.  Utilizes new Instance Attributes.


         14 October 2003  - Mark Quinsland  mark.quinsland@highqsoft.com
      Utilizes ASAM data types.



   -->
<!--
   *****************************************************************************************************
   *                                    Root Element of file                                          *
   *****************************************************************************************************
   -->
<atfx_file version="atfx_file v1.0.1" xmlns="http://www.asam.net"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.asam.net
odsbase_schema_5.0.rc5.xsd">
   <documentation>
      <exported_by>Mark Quinsland</exported_by>
      <exporter>Ascoba</exporter>
      <export_date_time>20.03.2003.12000000</export_date_time>
      <exporter_version>9.00.1234</exporter_version>
   </documentation>
   <!--
   *****************************************************************************************************
   *                                    Locale of Document                                            *
   *****************************************************************************************************
   -->
   <locale>US-EN</locale>
   <!--
```

```
*****************************************************************************************
*                        Based on ODS Base Model version 27                              *
*****************************************************************************************
-->
<base_model_version>27</base_model_version>
<!--
*****************************************************************************************
*                        declare any external files                                      *
*****************************************************************************************
-->
<files>
    <component>
        <identifier>file1</identifier>
        <filename>../data/k1.dat</filename>
    </component>
</files>
<!--
*****************************************************************************************
*                        declare application model meta data                             *
*****************************************************************************************
-->
<application_model>
    <application_element>
        <!--     ***  declare Engine Element  ***   -->
        <name>Engine</name>
        <basetype>AoTest</basetype>
        <application_attribute>
            <name>EngineName</name>
            <base_attribute>name</base_attribute>
            <unique>true</unique>
            <length>50</length>
        </application_attribute>
        <application_attribute>
            <name>EngineId</name>
            <base_attribute>id</base_attribute>
            <autogenerate>true</autogenerate>
            <obligatory>true</obligatory>
        </application_attribute>
        <application_attribute>
            <name>EngineVersion</name>
            <base_attribute>version</base_attribute>
            <obligatory>true</obligatory>
        </application_attribute>
        <application_attribute>
            <name>EngineDescription</name>
            <base_attribute>description</base_attribute>
            <length>100</length>
        </application_attribute>
        <application_attribute>
            <name>VersionDate</name>
            <base_attribute>version_date</base_attribute>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>CreateDate</name>
            <datatype>DT_DATE</datatype>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>bore</name>
            <datatype>DT_FLOAT</datatype>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>cylindernumber</name>
            <datatype>DT_SHORT</datatype>
            <obligatory>false</obligatory>
        </application_attribute>
        <relation_attribute>
            <name>SubTests</name>
            <ref_to>Test</ref_to>
            <base_relation>children</base_relation>
            <min_occurs>0</min_occurs>
            <max_occurs>Many</max_occurs>
        </relation_attribute>
        <relation_attribute>
            <name>EngineUser</name>
            <ref_to>User</ref_to>
            <min_occurs>0</min_occurs>
            <max_occurs>1</max_occurs>
        </relation_attribute>
    </application_element>
    <!--   ***  declare Test Element  ***    -->
    <application_element>
        <name>Test</name>
        <basetype>AoSubTest</basetype>
        <application_attribute>
            <name>TestName</name>
            <base_attribute>name</base_attribute>
            <unique>true</unique>
            <length>50</length>
        </application_attribute>
        <application_attribute>
            <name>TestId</name>
            <base_attribute>id</base_attribute>
            <autogenerate>true</autogenerate>
            <obligatory>true</obligatory>
        </application_attribute>
        <application_attribute>
            <name>TestVersion</name>
            <base_attribute>version</base_attribute>
            <obligatory>true</obligatory>
        </application_attribute>
        <application_attribute>
            <name>TestDescription</name>
            <base_attribute>description</base_attribute>
            <length>100</length>
        </application_attribute>
        <application_attribute>
            <name>VersionDate</name>
            <base_attribute>version_date</base_attribute>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
```

```
            <name>CreateDate</name>
            <datatype>DT_DATE</datatype>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>TestType</name>
            <datatype>DT_STRING</datatype>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>TestComment</name>
            <datatype>DT_STRING</datatype>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>exhaust</name>
            <datatype>DT_STRING</datatype>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>air_filter</name>
            <datatype>DT_STRING</datatype>
            <obligatory>false</obligatory>
        </application_attribute>
        <!--     ***  related elements  ***   -->
        <relation_attribute>
            <name>Measurements</name>
            <ref_to>Measurement</ref_to>
            <base_relation>children</base_relation>
            <min_occurs>0</min_occurs>
            <max_occurs>Many</max_occurs>
        </relation_attribute>
        <relation_attribute>
            <name>MainTest</name>
            <ref_to>Engine</ref_to>
            <base_relation>parent_test</base_relation>
            <min_occurs>0</min_occurs>
            <max_occurs>1</max_occurs>
        </relation_attribute>
        <relation_attribute>
            <name>TestUser</name>
            <ref_to>User</ref_to>
            <min_occurs>0</min_occurs>
            <max_occurs>1</max_occurs>
        </relation_attribute>
    </application_element>
    <!--   ***  declare Measurement Element  ***    -->
    <application_element>
        <name>Measurement</name>
        <basetype>AoMeasurement</basetype>
        <application_attribute>
            <name>MeaName</name>
            <base_attribute>name</base_attribute>
            <unique>true</unique>
            <length>50</length>
        </application_attribute>
        <application_attribute>
        <name>MeaId</name>
        <base_attribute>id</base_attribute>
        <autogenerate>true</autogenerate>
        <obligatory>true</obligatory>
    </application_attribute>
    <application_attribute>
        <name>MeaVersion</name>
        <base_attribute>version</base_attribute>
        <obligatory>true</obligatory>
    </application_attribute>
    <application_attribute>
        <name>MeaDescription</name>
        <base_attribute>description</base_attribute>
        <length>100</length>
    </application_attribute>
    <application_attribute>
        <name>VersionDate</name>
        <base_attribute>version_date</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>MeasurementBegin</name>
        <base_attribute>measurement_begin</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>MeasurementEnd</name>
        <base_attribute>measurement_end</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>CreateDate</name>
        <datatype>DT_DATE</datatype>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>MeaType</name>
        <datatype>DT_STRING</datatype>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>MeaComment</name>
        <datatype>DT_STRING</datatype>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>ROZ</name>
        <datatype>DT_FLOAT</datatype>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>air_filter</name>
        <datatype>DT_STRING</datatype>
        <obligatory>false</obligatory>
    </application_attribute>
    <!--     ***  related elements  ***   -->
    <relation_attribute>
```

```
            <name>Test</name>
            <ref_to>Test</ref_to>
            <base_relation>test</base_relation>
            <min_occurs>0</min_occurs>
            <max_occurs>Many</max_occurs>
        </relation_attribute>
        <relation_attribute>
            <name>MeaQuantities</name>
            <ref_to>MeasurmentQuantity</ref_to>
            <base_relation>measurement_quantities</base_relation>
            <min_occurs>0</min_occurs>
            <max_occurs>1</max_occurs>
        </relation_attribute>
        <relation_attribute>
            <name>Submatrices</name>
            <ref_to>Submatrix</ref_to>
            <base_relation>submatrices</base_relation>
            <min_occurs>0</min_occurs>
            <max_occurs>1</max_occurs>
        </relation_attribute>
    </application_element>
    <!--   ***  declare Measurement Quantity Element  *** -->
    <application_element>
        <name>MeasurementQuantity</name>
        <basetype>AoMeasurementQuantity</basetype>
        <application_attribute>
            <name>MeaQName</name>
            <base_attribute>name</base_attribute>
            <unique>true</unique>
            <length>50</length>
        </application_attribute>
        <application_attribute>
            <name>MeaQId</name>
            <base_attribute>id</base_attribute>
            <autogenerate>true</autogenerate>
            <obligatory>true</obligatory>
        </application_attribute>
        <application_attribute>
            <name>MeaQVersion</name>
            <base_attribute>version</base_attribute>
            <obligatory>true</obligatory>
        </application_attribute>
        <application_attribute>
            <name>MeaQDescription</name>
            <base_attribute>description</base_attribute>
            <length>100</length>
        </application_attribute>
        <application_attribute>
            <name>VersionDate</name>
            <base_attribute>version_date</base_attribute>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>Rank</name>
            <base_attribute>rank</base_attribute>
            <obligatory>false</obligatory>
        </application_attribute>
```

```
<application_attribute>
    <name>Dimension</name>
    <base_attribute>dimension</base_attribute>
    <obligatory>false</obligatory>
</application_attribute>
<application_attribute>
    <name>DataType</name>
    <base_attribute>datatype_enum</base_attribute>
    <obligatory>false</obligatory>
</application_attribute>
<application_attribute>
    <name>TypeSize</name>
    <base_attribute>type_size</base_attribute>
    <obligatory>false</obligatory>
</application_attribute>
<application_attribute>
    <name>Interpolation</name>
    <base_attribute>interpolation</base_attribute>
    <obligatory>false</obligatory>
</application_attribute>
<application_attribute>
    <name>Minimum</name>
    <base_attribute>minimum</base_attribute>
    <obligatory>false</obligatory>
</application_attribute>
<application_attribute>
    <name>Maximum</name>
    <base_attribute>maximum</base_attribute>
    <obligatory>false</obligatory>
</application_attribute>
<application_attribute>
    <name>Average</name>
    <base_attribute>average</base_attribute>
    <obligatory>false</obligatory>
</application_attribute>
<application_attribute>
    <name>SDeviation</name>
    <base_attribute>standard_deviation</base_attribute>
    <obligatory>false</obligatory>
</application_attribute>
<application_attribute>
    <name>MeaQuantStat</name>
    <datatype>DT_SHORT</datatype>
    <obligatory>false</obligatory>
</application_attribute>
<application_attribute>
    <name>MeaQuantStat</name>
    <datatype>DT_SHORT</datatype>
    <obligatory>false</obligatory>
</application_attribute>
<application_attribute>
    <name>TimeOffset</name>
    <datatype>DT_DOUBLE</datatype>
    <obligatory>false</obligatory>
</application_attribute>
<application_attribute>
    <name>SamplingRate</name>
```

```
            <datatype>DT_DOUBLE</datatype>
            <obligatory>false</obligatory>
        </application_attribute>
        <!--     ***  related elements  ***   -->
        <relation_attribute>
            <name>Measurement</name>
            <ref_to>measurement</ref_to>
            <base_relation>measurement</base_relation>
        </relation_attribute>
        <relation_attribute>
            <name>LocalColumns</name>
            <ref_to>LocalColumn</ref_to>
            <base_relation>local_columns</base_relation>
            <min_occurs>0</min_occurs>
            <max_occurs>Many</max_occurs>
        </relation_attribute>
        <relation_attribute>
            <name>Quantity</name>
            <ref_to>Quantity</ref_to>
            <base_relation>quantity</base_relation>
        </relation_attribute>
        <relation_attribute>
            <name>Unit</name>
            <ref_to>Unit</ref_to>
            <base_relation>unit</base_relation>
        </relation_attribute>
    </application_element>
    <!--   ***  declare SubmatrixElement  ***     -->
    <application_element>
        <name>Submatrix</name>
        <basetype>AoSubmatrix</basetype>
        <application_attribute>
            <name>name</name>
            <base_attribute>name</base_attribute>
            <length>50</length>
        </application_attribute>
        <application_attribute>
            <name>id</name>
            <base_attribute>id</base_attribute>
            <autogenerate>true</autogenerate>
            <obligatory>true</obligatory>
        </application_attribute>
        <application_attribute>
            <name>version</name>
            <base_attribute>version</base_attribute>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>description</name>
            <base_attribute>description</base_attribute>
            <obligatory>false</obligatory>
            <length>100</length>
        </application_attribute>
        <application_attribute>
            <name>version_date</name>
            <base_attribute>version_date</base_attribute>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>Rank</name>
            <base_attribute>rank</base_attribute>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>number_of_rows</name>
            <base_attribute>number_of_rows</base_attribute>
            <obligatory>false</obligatory>
        </application_attribute>
        <!--     ***  related elements  ***   -->
        <relation_attribute>
            <name>Measurement</name>
            <ref_to>measurement</ref_to>
            <base_relation>measurement</base_relation>
        </relation_attribute>
        <relation_attribute>
            <name>LocalColumns</name>
            <ref_to>LocalColumn</ref_to>
            <base_relation>local_columns</base_relation>
            <min_occurs>0</min_occurs>
            <max_occurs>Many</max_occurs>
        </relation_attribute>
    </application_element>
    <!--   ***  declare Local Column Element  ***    -->
    <application_element>
        <name>LocalColumn</name>
        <basetype>AoLocalColumn</basetype>
        <application_attribute>
            <name>name</name>
            <base_attribute>name</base_attribute>
            <length>50</length>
        </application_attribute>
        <application_attribute>
            <name>id</name>
            <base_attribute>id</base_attribute>
            <autogenerate>true</autogenerate>
            <obligatory>true</obligatory>
        </application_attribute>
        <application_attribute>
            <name>version</name>
            <base_attribute>version</base_attribute>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>description</name>
            <base_attribute>description</base_attribute>
            <obligatory>false</obligatory>
            <length>100</length>
        </application_attribute>
        <application_attribute>
            <name>version_date</name>
            <base_attribute>version_date</base_attribute>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
        <name>Rank</name>
        <base_attribute>rank</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>flags</name>
        <base_attribute>flags</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>global_flag</name>
        <base_attribute>global_flag</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>independent</name>
        <base_attribute>independent</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>minimum</name>
        <base_attribute>minimum</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>maximum</name>
        <base_attribute>maximum</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>values</name>
        <base_attribute>values_float64</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <!--     ***  related elements  ***   -->
    <relation_attribute>
        <name>submatrix</name>
        <ref_to>Submatrix</ref_to>
        <base_relation>measurement</base_relation>
    </relation_attribute>
    <relation_attribute>
        <name>measurement_quantity</name>
        <ref_to>MeaQuantity</ref_to>
        <base_relation>measurement_quantity</base_relation>
    </relation_attribute>
</application_element>
<!--   ***  declare Quantity Element  ***   -->
<application_element>
    <name>Quantity</name>
    <basetype>AoQuantity</basetype>
    <application_attribute>
        <name>QName</name>
        <base_attribute>name</base_attribute>
        <length>50</length>
    </application_attribute>
    <application_attribute>
        <name>QId</name>
        <base_attribute>id</base_attribute>
        <autogenerate>true</autogenerate>
        <obligatory>true</obligatory>
    </application_attribute>
    <application_attribute>
        <name>QVersion</name>
        <base_attribute>version</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>QDescription</name>
        <base_attribute>description</base_attribute>
        <obligatory>false</obligatory>
        <length>100</length>
    </application_attribute>
    <application_attribute>
        <name>version_date</name>
        <base_attribute>version_date</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>DefaultRank</name>
        <base_attribute>default_rank</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>DefaultDimension</name>
        <base_attribute>default_dimension</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>DefaultDataType</name>
        <base_attribute>default_datatype</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>DefaultType_size</name>
        <base_attribute>default_type_size</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>DefaultMQName</name>
        <base_attribute>default_mq_name</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <!--     ***  related elements  ***   -->
    <relation_attribute>
        <name>DefaultUnit</name>
        <ref_to>Unit</ref_to>
        <base_relation>default_unit</base_relation>
    </relation_attribute>
    <relation_attribute>
        <name>Successors</name>
        <ref_to>Quantity</ref_to>
        <base_relation>successors</base_relation>
        <min_occurs>0</min_occurs>
        <max_occurs>Many</max_occurs>
        </relation_attribute>
        <relation_attribute>
            <name>Predeccessors</name>
            <ref_to>Quantity</ref_to>
            <base_relation>predecessor</base_relation>
        </relation_attribute>
        <relation_attribute>
            <name>QuantityGroup</name>
            <ref_to>QuantityGroup</ref_to>
            <base_relation>groups</base_relation>
            <min_occurs>0</min_occurs>
            <max_occurs>Many</max_occurs>
        </relation_attribute>
    </application_element>
    <!--   ***  declare QuantityGroup Element  ***   -->
    <application_element>
        <name>QuantityGroup</name>
        <basetype>AoQuantityGroup</basetype>
        <application_attribute>
            <name>QGroupName</name>
            <base_attribute>name</base_attribute>
            <length>50</length>
        </application_attribute>
        <application_attribute>
            <name>QGroupId</name>
            <base_attribute>id</base_attribute>
            <autogenerate>true</autogenerate>
            <obligatory>true</obligatory>
        </application_attribute>
        <application_attribute>
            <name>QGroupVersion</name>
            <base_attribute>version</base_attribute>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>QGroupDescription</name>
            <base_attribute>description</base_attribute>
            <obligatory>false</obligatory>
            <length>100</length>
        </application_attribute>
        <application_attribute>
            <name>version_date</name>
            <base_attribute>version_date</base_attribute>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>QuantGroupState</name>
            <datatype>DT_SHORT</datatype>
            <obligatory>false</obligatory>
        </application_attribute>
        <!--      ***  related elements  ***   -->
        <relation_attribute>
            <name>ListOfQuantities</name>
            <ref_to>Quantity</ref_to>
            <base_relation>quantities</base_relation>
            <min_occurs>0</min_occurs>
            <max_occurs>Many</max_occurs>
    </relation_attribute>
</application_element>
<!--   ***  declare Unit Element  ***    -->
<application_element>
    <name>Unit</name>
    <basetype>AoUnit</basetype>
    <application_attribute>
        <name>UnitName</name>
        <base_attribute>name</base_attribute>
        <length>50</length>
    </application_attribute>
    <application_attribute>
        <name>UnitId</name>
        <base_attribute>id</base_attribute>
        <autogenerate>true</autogenerate>
        <obligatory>true</obligatory>
    </application_attribute>
    <application_attribute>
        <name>UnitVersion</name>
        <base_attribute>version</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>UnitDescription</name>
        <base_attribute>description</base_attribute>
        <obligatory>false</obligatory>
        <length>100</length>
    </application_attribute>
    <application_attribute>
        <name>version_date</name>
        <base_attribute>version_date</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>UnitFactor</name>
        <base_attribute>factor</base_attribute>
        <obligatory>true</obligatory>
    </application_attribute>
    <application_attribute>
        <name>UnitOffset</name>
        <base_attribute>offset</base_attribute>
        <obligatory>true</obligatory>
    </application_attribute>
    <application_attribute>
        <name>QuantGroupState</name>
        <datatype>DT_SHORT</datatype>
        <obligatory>false</obligatory>
    </application_attribute>
    <!--      ***  related elements  ***    -->
    <relation_attribute>
        <name>PhysDim</name>
        <ref_to>PhysDim</ref_to>
        <base_relation>phys_dimension</base_relation>
    </relation_attribute>
    <relation_attribute>
        <name>UnitGroups</name>
        <ref_to>UnitGroup</ref_to>
        <base_relation>groups</base_relation>
        <min_occurs>0</min_occurs>
        <max_occurs>Many</max_occurs>
    </relation_attribute>
</application_element>
<!--   ***  declare Unit Group Element  ***    -->
<application_element>
    <name>UnitGroup</name>
    <basetype>AoUnit</basetype>
    <application_attribute>
        <name>UnitName</name>
        <base_attribute>name</base_attribute>
        <length>50</length>
    </application_attribute>
    <application_attribute>
        <name>UnitGroupId</name>
        <base_attribute>id</base_attribute>
        <autogenerate>true</autogenerate>
        <obligatory>true</obligatory>
    </application_attribute>
    <application_attribute>
        <name>UnitGroupVersion</name>
        <base_attribute>version</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>UnitGroupDescription</name>
        <base_attribute>description</base_attribute>
        <obligatory>false</obligatory>
        <length>100</length>
    </application_attribute>
    <application_attribute>
        <name>version_date</name>
        <base_attribute>version_date</base_attribute>
        <obligatory>false</obligatory>
    </application_attribute>
    <application_attribute>
        <name>UnitGroupState</name>
        <datatype>DT_SHORT</datatype>
        <obligatory>true</obligatory>
    </application_attribute>
    <!--      ***  related elements  ***    -->
    <relation_attribute>
        <name>PhysDim</name>
        <ref_to>PhysDim</ref_to>
        <base_relation>phys_dimension</base_relation>
    </relation_attribute>
    <relation_attribute>
        <name>Units</name>
        <ref_to>Unit</ref_to>
        <base_relation>units</base_relation>
        <min_occurs>0</min_occurs>
        <max_occurs>Many</max_occurs>
    </relation_attribute>
</application_element>
<!--   ***  declare Physical Dimension Element  ***      -->
<application_element>
```

```
<name>PhysDim</name>
<basetype>AoPhysicalDimension</basetype>
<application_attribute>
    <name>PhysDimName</name>
    <base_attribute>name</base_attribute>
    <length>50</length>
</application_attribute>
<application_attribute>
    <name>PhysDimId</name>
    <base_attribute>id</base_attribute>
    <autogenerate>true</autogenerate>
    <obligatory>true</obligatory>
</application_attribute>
<application_attribute>
    <name>PhysDimVersion</name>
    <base_attribute>version</base_attribute>
    <obligatory>false</obligatory>
</application_attribute>
<application_attribute>
    <name>PhysDimDescription</name>
    <base_attribute>description</base_attribute>
    <obligatory>false</obligatory>
    <length>100</length>
</application_attribute>
<application_attribute>
    <name>version_date</name>
    <base_attribute>version_date</base_attribute>
    <obligatory>false</obligatory>
</application_attribute>
<application_attribute>
    <name>length</name>
    <base_attribute>length_exp</base_attribute>
    <obligatory>true</obligatory>
</application_attribute>
<application_attribute>
    <name>mass</name>
    <base_attribute>mass_exp</base_attribute>
    <obligatory>true</obligatory>
</application_attribute>
<application_attribute>
    <name>time</name>
    <base_attribute>time_exp</base_attribute>
    <obligatory>true</obligatory>
</application_attribute>
<application_attribute>
    <name>current</name>
    <base_attribute>current_exp</base_attribute>
    <obligatory>true</obligatory>
</application_attribute>
<application_attribute>
    <name>temperature</name>
    <base_attribute>temperature_exp</base_attribute>
    <obligatory>true</obligatory>
</application_attribute>
<application_attribute>
    <name>molar</name>
    <base_attribute>molar_amount_exp</base_attribute>
```

```
            <obligatory>true</obligatory>
        </application_attribute>
        <application_attribute>
            <name>luminous</name>
            <base_attribute>luminous_intensity_exp</base_attribute>
            <obligatory>true</obligatory>
        </application_attribute>
        <!--     ***  related elements  ***   -->
        <relation_attribute>
            <name>Units</name>
            <ref_to>Unit</ref_to>
            <base_relation>units</base_relation>
            <min_occurs>0</min_occurs>
            <max_occurs>Many</max_occurs>
        </relation_attribute>
    </application_element>
    <!--   ***  declare User Element for Security Information ***    -->
    <application_element>
        <name>User</name>
        <basetype>AoUser</basetype>
        <application_attribute>
            <name>UserName</name>
            <base_attribute>name</base_attribute>
            <length>50</length>
        </application_attribute>
        <application_attribute>
            <name>UserId</name>
            <base_attribute>id</base_attribute>
            <autogenerate>true</autogenerate>
            <obligatory>true</obligatory>
        </application_attribute>
        <application_attribute>
            <name>UserVersion</name>
            <base_attribute>version</base_attribute>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>UserDescription</name>
            <base_attribute>description</base_attribute>
            <obligatory>false</obligatory>
            <length>100</length>
        </application_attribute>
        <application_attribute>
            <name>version_date</name>
            <base_attribute>version_date</base_attribute>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>password</name>
            <base_attribute>password</base_attribute>
            <obligatory>true</obligatory>
        </application_attribute>
        <application_attribute>
            <name>UserDepartment</name>
            <datatype>DT_STRING</datatype>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>AliasName</name>
            <datatype>DT_STRING</datatype>
            <obligatory>false</obligatory>
        </application_attribute>
        <!--    ***  related elements  ***   -->
        <relation_attribute>
            <name>groups</name>
            <ref_to>UserGroup</ref_to>
            <base_relation>groups</base_relation>
            <min_occurs>0</min_occurs>
            <max_occurs>Many</max_occurs>
        </relation_attribute>
    </application_element>
    <!--   ***  declare User Group Element for Security Information ***          -->
    <application_element>
        <name>UserGroup</name>
        <basetype>AoUserGroup</basetype>
        <application_attribute>
            <name>UserGroupName</name>
            <base_attribute>name</base_attribute>
            <length>50</length>
        </application_attribute>
        <application_attribute>
            <name>UserGroupId</name>
            <base_attribute>id</base_attribute>
            <autogenerate>true</autogenerate>
            <obligatory>true</obligatory>
        </application_attribute>
        <application_attribute>
            <name>UserGroupVersion</name>
            <base_attribute>version</base_attribute>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>UserGroupDescription</name>
            <base_attribute>description</base_attribute>
            <obligatory>false</obligatory>
            <length>100</length>
        </application_attribute>
        <application_attribute>
            <name>version_date</name>
            <base_attribute>version_date</base_attribute>
            <obligatory>false</obligatory>
        </application_attribute>
        <application_attribute>
            <name>superuser_flag</name>
            <base_attribute>superuser_flag</base_attribute>
            <obligatory>true</obligatory>
        </application_attribute>
        <!--    ***  related elements  ***   -->
        <relation_attribute>
            <name>users</name>
            <ref_to>User</ref_to>
            <base_relation>users</base_relation>
            <min_occurs>0</min_occurs>
            <max_occurs>Many</max_occurs>
        </relation_attribute>
    </application_element>
    <!--    ***  declare Access Control List Element for Security Information ***            -->
    <application_element>
        <name>ACLA</name>
        <basetype>ACLA</basetype>
        <application_attribute>
            <name>appl_element_id</name>
            <base_attribute>appl_element_id</base_attribute>
            <obligatory>true</obligatory>
        </application_attribute>
        <application_attribute>
            <name>rights</name>
            <base_attribute>rights</base_attribute>
            <obligatory>true</obligatory>
        </application_attribute>
        <application_attribute>
            <name>attribute_name</name>
            <base_attribute>attribute_name</base_attribute>
            <obligatory>true</obligatory>
        </application_attribute>
        <!--      ***  related elements  ***   -->
        <relation_attribute>
            <name>users</name>
            <ref_to>User</ref_to>
            <base_relation>users</base_relation>
        </relation_attribute>
    </application_element>
    <!--    ***  declare Access Control List Element for Security Information ***            -->
    <application_element>
        <name>ACLI</name>
        <basetype>ACLA</basetype>
        <application_attribute>
            <name>appl_element_id</name>
            <base_attribute>appl_element_id</base_attribute>
            <obligatory>true</obligatory>
        </application_attribute>
        <application_attribute>
            <name>rights</name>
            <base_attribute>rights</base_attribute>
            <obligatory>true</obligatory>
        </application_attribute>
        <application_attribute>
            <name>instance_id</name>
            <base_attribute>instance_id</base_attribute>
            <obligatory>true</obligatory>
        </application_attribute>
        <!--      ***  related elements  ***   -->
        <relation_attribute>
            <name>users</name>
            <ref_to>User</ref_to>
            <base_relation>users</base_relation>
        </relation_attribute>
    </application_element>
    <!--    ***  Example of Application Enumeration***    -->
    <application_enumeration>
        <name>MyEnum</name>
        <item>
            <name>first</name>
            <value>1</value>
        </item>
        <item>
            <name>second</name>
            <value>2</value>
        </item>
    </application_enumeration>
</application_model>
<!--
*****************************************************************************************
*                              end application model meta data                          *
*****************************************************************************************
-->
<!--
*****************************************************************************************
*                              Instance Data is NOT validated with Base Schema          *
*****************************************************************************************
-->
<instance_data>
<!--
*****************************************************************************************
*   Instance elements of the application elements Engine and Test
*   from base type AoTest and AoSubTest
*****************************************************************************************
-->
    <Engine>
        <EngineName>Test Engine 1</EngineName>
        <EngineId>1</EngineId>
        <EngineVersion>A 1.0</EngineVersion>
        <EngineDescription>The first test engine</EngineDescription>
        <VersionDate>199702010900</VersionDate>
        <CreateDate>199702010900</CreateDate>
        <EngineType>ABC47111</EngineType>
        <bore>92</bore>
        <cylindernumber>6</cylindernumber>
        <stroke>95.321</stroke>
        <SubTests>1</SubTests>
        <EngineUser>21</EngineUser>
    </Engine>
    <Test>
        <TestName>Test-Configuration 1</TestName>
        <TestId>1</TestId>
        <TestVersion>B20</TestVersion>
        <TestDescription>The first test of engine 1</TestDescription>
        <VersionDate>199702010900</VersionDate>
        <CreateDate>199702010900</CreateDate>
        <TestType>Functionality Test</TestType>
        <TestComment>Test: torque and at special points temperature</TestComment>
        <exhaust>System 4711</exhaust>
        <air_filter>C 12.1</air_filter>
        <!--  Ref to instance elements of Measurements  -->
        <Measurements>1 12</Measurements>
        <!--  Ref to instance element of Engine  -->
        <MainTest>1 </MainTest>
        <!--  Ref to instance element of User  -->
        <TestUser>22</TestUser>
    </Test>
<!--
******************************************************************************************
*   Instance elements of the application model part with
*   Measurement, MeaQuantity, Submatrix, LocalColumn with value_sequence
*   for the first measurement
******************************************************************************************
-->
    <Measurement>
        <MeaName>M001A</MeaName>
        <MeaId>1</MeaId>
        <MeaVersion>A1</MeaVersion>
        <MeaDescription>1. Measurement</MeaDescription>
        <VersionDate>199702010900</VersionDate>
        <CreateDate>199702010900</CreateDate>
        <MeasurementBegin>19971210141345</MeasurementBegin>
        <MeasurementEnd>199712100162135</MeasurementEnd>
        <MeaType>Functionality Test</MeaType>
        <MeaComment>No Problem</MeaComment>
        <ROZ/>
        <!--  Ref to instance elements of Test  -->
        <Test>1</Test>
        <!--  Ref to instance elements of MeaQuantity  -->
        <MeaQuantities>1 2 3</MeaQuantities>
        <!--  Ref to instance element of Submatrix -->
        <Submatrices>1</Submatrices>
    </Measurement>
    <MeaQuantity>
        <MeaQName>N</MeaQName>
        <MeaQId>1</MeaQId>
        <MeaQVersion>D1</MeaQVersion>
        <MeaQDescription>Number of revolutions</MeaQDescription>
        <version_date/>
        <Rank>0</Rank>
        <Dimension/>
        <Interpolation/>
        <Minimum>500</Minimum>
        <Maximum>3200</Maximum>
        <Average>1850</Average>
        <SDeviation/>
        <TimeOffset/>
        <SamplingRate/>
        <Measurement>1</Measurement>
        <LocalColumns>1</LocalColumns>
        <Quantity>5</Quantity>
        <Unit>4</Unit>
    </MeaQuantity>
    <MeaQuantity>
        <MeaQName>MD</MeaQName>
        <MeaQId>2</MeaQId>
        <MeaQVersion>D</MeaQVersion>
        <MeaQDescription>Torque</MeaQDescription>
        <version_date/>
        <Rank>0</Rank>
        <Dimension/>
        <DataType>DT_FLOAT</DataType>
    <Interpolation/>
    <Minimum>150.</Minimum>
    <Maximum>255.</Maximum>
    <Average>190.</Average>
    <SDeviation>22.2</SDeviation>
    <TimeOffset/>
    <SamplingRate/>
    <Measurement>1</Measurement>
    <LocalColumns>2</LocalColumns>
    <Quantity>6</Quantity>
    <Unit>6</Unit>
</MeaQuantity>
<MeaQuantity>
    <MeaQName>PL</MeaQName>
    <MeaQId>3</MeaQId>
    <MeaQVersion>D</MeaQVersion>
    <MeaQDescription>Pressure</MeaQDescription>
    <version_date/>
    <Rank>0</Rank>
    <Dimension/>
    <DataType>DT_FLOAT</DataType>
    <Interpolation/>
    <Minimum>991.2</Minimum>
    <Maximum>993.2</Maximum>
    <Average>992.0</Average>
    <SDeviation/>
    <TimeOffset/>
    <SamplingRate/>
    <Measurement>1</Measurement>
    <LocalColumns>3</LocalColumns>
    <Quantity>7</Quantity>
    <Unit>7</Unit>
</MeaQuantity>
<Submatrix>
    <name>M001A</name>
    <id>1</id>
    <version>1a</version>
    <description>1. Measurement for data transfer via ASAM</description>
    <version_date/>
    <number_of_rows>15</number_of_rows>
    <measurement>1</measurement>
    <local_columns>1 2 3</local_columns>
</Submatrix>
<LocalColumn>
    <name>N</name>
    <id>1</id>
    <version/>
    <description/>
    <version_date/>
    <flags>0</flags>
    <global_flag>0</global_flag>
    <independent>true</independent>
    <minimum>500</minimum>
    <maximum>3200.</maximum>
    <datatype>DT_FLOAT</datatype>
    <values>
        500. 700. 900. 1100. 1300. 1400. 1600. 1800. 2000. 2200. 2400. 2600. 2800. 3000. 3200.
```

```
            </values>
            <submatrix>1</submatrix>
            <measurement_quantity>1</measurement_quantity>
        </LocalColumn>
        <LocalColumn>
            <name>MD</name>
            <id>2</id>
            <version/>
            <description/>
            <version_date/>
            <flags>0</flags>
            <global_flag>0</global_flag>
            <independent>false</independent>
            <minimum>150</minimum>
            <maximum>255.</maximum>
            <DataType>DT_FLOAT</DataType>
            <values>
        150. 160. 170. 180. 190. 200. 210. 220 230. 240. 250. 255. 250. 230. 190
    </values>
            <submatrix>1</submatrix>
            <measurement_quantity>2</measurement_quantity>
        </LocalColumn>
        <LocalColumn>
            <name>PL</name>
            <id>3</id>
            <version/>
            <description/>
            <version_date/>
            <flags>0</flags>
            <global_flag>0</global_flag>
            <independent>false</independent>
            <minimum>991.2</minimum>
            <maximum>993.2</maximum>
            <DataType>DT_FLOAT</DataType>
            <values>
              991.2 991.7 991.5 991.9 992.1 987.6 992.2 993.0 993.2 993.2 992.7 993.0 993.0 992.7 992.7
            </values>
            <submatrix>1</submatrix>
            <measurement_quantity>3</measurement_quantity>
        </LocalColumn>
    <!--
    ************************************************************************************************
    *    Instance elements of the application model part with
    *    Measurement, MeaQuantity, Submatrix, LocalColumn with value_sequence
    *    for the second measurement
    ************************************************************************************************
    -->
        <Measurement>
            <MeaName>M002A</MeaName>
            <MeaId>2</MeaId>
            <MeaVersion>A1</MeaVersion>
            <MeaDescription>2. Measurement</MeaDescription>
            <version_date>199702010900</version_date>
            <MeasurementBegin>19971211141345</MeasurementBegin>
            <MeasurementEnd>199712110162135</MeasurementEnd>
            <CreateDate>199712110162135</CreateDate>
            <MeaType>VLM</MeaType>
        <MeaComment>No Problem</MeaComment>
        <ROZ/>
        <!--   Ref to instance elements of Test  -->
        <Test>1</Test>
        <!--   Ref to instance elements of MeaQuantity  -->
        <MeaQuantities>4 5 6 7 </MeaQuantities>
        <!--   Ref to instance element of Submatrix -->
        <Submatrices>2 3</Submatrices>
    </Measurement>
    <MeaQuantity>
        <MeaQName>SPTNR</MeaQName>
        <MeaQId>4</MeaQId>
        <MeaQVersion>A</MeaQVersion>
        <MeaQDescription>System point number</MeaQDescription>
        <version_date/>
        <Rank>0</Rank>
        <Dimension/>
        <Interpolation/>
        <Minimum>1.0</Minimum>
        <Maximum>15</Maximum>
        <Average>7.5</Average>
        <SDeviation/>
        <TimeOffset/>
        <SamplingRate/>
        <Measurement>2</Measurement>
        <LocalColumns>4 7</LocalColumns>
        <Quantity>4</Quantity>
        <Unit>1</Unit>
    </MeaQuantity>
    <MeaQuantity>
        <MeaQName>N</MeaQName>
        <MeaQId>5</MeaQId>
        <MeaQVersion>D</MeaQVersion>
        <MeaQDescription>Revolution</MeaQDescription>
        <version_date/>
        <Rank>0</Rank>
        <Dimension/>
        <DataType>DT_FLOAT</DataType>
        <Interpolation/>
        <Minimum>500</Minimum>
        <Maximum>3200</Maximum>
        <Average>1850</Average>
        <SDeviation/>
        <TimeOffset/>
        <SamplingRate/>
        <Measurement>2</Measurement>
        <LocalColumns>5</LocalColumns>
        <Quantity>5</Quantity>
        <Unit>4</Unit>
    </MeaQuantity>
    <MeaQuantity>
        <MeaQName>MD</MeaQName>
        <MeaQId>6</MeaQId>
        <MeaQVersion>D</MeaQVersion>
        <MeaQDescription>Torque</MeaQDescription>
        <version_date/>
        <Rank>0</Rank>
```

```
          <Dimension/>
          <DataType>DT_FLOAT</DataType>
          <Interpolation/>
          <Minimum>152.</Minimum>
          <Maximum>257.</Maximum>
          <Average>194.</Average>
          <SDeviation/>
          <TimeOffset/>
          <SamplingRate/>
          <Measurement>2</Measurement>
          <LocalColumns>6</LocalColumns>
          <Quantity>6</Quantity>
          <Unit>6</Unit>
      </MeaQuantity>
      <MeaQuantity>
          <MeaQName>TEMP1</MeaQName>
          <MeaQId>7</MeaQId>
          <MeaQVersion>D</MeaQVersion>
          <MeaQDescription>Temperature</MeaQDescription>
          <version_date/>
          <Rank>0</Rank>
          <Dimension/>
          <DataType>DT_FLOAT</DataType>
          <Interpolation/>
          <Minimum>150.</Minimum>
          <Maximum>183.</Maximum>
          <Average>169.</Average>
          <SDeviation/>
          <TimeOffset/>
          <SamplingRate/>
          <Measurement>2</Measurement>
          <LocalColumns>8</LocalColumns>
          <Quantity>2</Quantity>
          <Unit>2</Unit>
      </MeaQuantity>
      <Submatrix>
          <name>M002A</name>
          <id>2</id>
          <version>1a</version>
          <description>2. Measurement for data transfer via ASAM</description>
          <version_date/>
          <number_of_rows>15</number_of_rows>
          <measurement>2</measurement>
          <local_columns>4 5 6</local_columns>
      </Submatrix>
      <LocalColumn>
          <name>SPKTNR</name>
          <id>4</id>
          <version/>
          <description/>
          <version_date/>
          <flags>0</flags>
          <global_flag>0</global_flag>
          <independent>true</independent>
          <minimum>1</minimum>
          <maximum>15</maximum>
          <DataType>DT_SHORT</DataType>
          <values>
      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
  </values>
          <submatrix>2</submatrix>
          <measurement_quantity>4</measurement_quantity>
      </LocalColumn>
      <LocalColumn>
          <name>N</name>
          <id>5</id>
          <version/>
          <description/>
          <version_date/>
          <flags>0</flags>
          <global_flag>0</global_flag>
          <independent>false</independent>
          <minimum>500</minimum>
          <maximum>3200.</maximum>
          <DataType>DT_SHORT</DataType>
          <values>
      500. 700. 900. 1100. 1300. 1400. 1600. 1800. 2000. 2200. 2400. 2600. 2800. 3000. 3200.
  </values>
          <submatrix>2</submatrix>
          <measurement_quantity>5</measurement_quantity>
      </LocalColumn>
      <LocalColumn>
          <name>MD</name>
          <id>6</id>
          <version/>
          <description/>
          <version_date/>
          <flags>0</flags>
          <global_flag>0</global_flag>
          <independent>false</independent>
          <minimum>150</minimum>
          <maximum>255.</maximum>
          <DataType>DT_FLOAT</DataType>
          <values>
              150. 160. 170. 180. 190. 200. 210. 220 230. 240. 250. 255. 250. 230. 190
          </values>
          <submatrix>2</submatrix>
          <measurement_quantity>6</measurement_quantity>
      </LocalColumn>
      <Submatrix>
          <name>M002B</name>
          <id>3</id>
          <version>1a</version>
          <description>2. Measurement for data transfer via ASAM</description>
          <version_date/>
          <number_of_rows>3</number_of_rows>
          <measurement>2</measurement>
          <local_columns>7 8</local_columns>
      </Submatrix>
      <LocalColumn>
          <name>SPKTNR</name>
          <id>7</id>
          <version/>
          <description/>
```

```
            <version_date/>
            <flags>0</flags>
            <global_flag>0</global_flag>
            <independent>true</independent>
            <minimum>5</minimum>
            <maximum>15</maximum>
            <DataType>DT_SHORT</DataType>
            <values>
            5 10 15
   </values>
            <submatrix>2</submatrix>
            <measurement_quantity>4</measurement_quantity>
        </LocalColumn>
        <LocalColumn>
            <name>TEMP1</name>
            <id>8</id>
            <version/>
            <description/>
            <version_date/>
            <flags>0</flags>
            <global_flag>0</global_flag>
            <independent>false</independent>
            <minimum>150</minimum>
            <maximum>183</maximum>
            <DataType>DT_FLOAT</DataType>
            <values>
            150. 174. 183.
            </values>
            <submatrix>2</submatrix>
            <measurement_quantity>7</measurement_quantity>
        </LocalColumn>
    <!--
    ****************************************************************************************************
    *   instance elements of the application elements User, User Group
    ****************************************************************************************************
    -->
        <User>
            <UserName>Peter Sellers</UserName>
            <UserId>21</UserId>
            <UserVersion>1A</UserVersion>
            <UserDescription>Super user group 1</UserDescription>
            <version_date>199708101105</version_date>
            <password>^1kjws7hxoish^mjkkdjxo"e--@@@kjg798xh0880djdj90</password>
            <groups>30 50</groups>
            <UserDepartment>OP/EFG</UserDepartment>
            <AliasName>PS</AliasName>
        </User>
        <User>
            <UserName>Todd Martin</UserName>
            <UserId>22</UserId>
            <UserVersion>1B</UserVersion>
            <UserDescription>Sub user group 1b</UserDescription>
            <version_date>199708101105</version_date>
            <password>j</password>
            <groups>30 40</groups>
            <UserDepartment>OP/ABC</UserDepartment>
            <AliasName>TM</AliasName>
    </User>
    <User>
        <UserName>Otto Bierman</UserName>
        <UserId>23</UserId>
        <UserVersion>1A</UserVersion>
        <UserDescription>Sub user group 1</UserDescription>
        <version_date>199708101105</version_date>
        <password>89xzhuon3m,”))?imi/U(/%unpo39048ejdkádksos••o•3</password>
        <groups>40 50</groups>
        <UserDepartment>OP/EFG</UserDepartment>
        <AliasName>OB</AliasName>
    </User>
    <UserGroup>
        <GroupName>AQ 1</GroupName>
        <GroupId>30</GroupId>
        <GroupVersion>00</GroupVersion>
        <GroupDescription>Group1</GroupDescription>
        <version_date>199708101105</version_date>
        <superuser_flag>true</superuser_flag>
        <users>21 22</users>
    </UserGroup>
    <UserGroup>
        <GroupName>AQ 2</GroupName>
        <GroupId>40</GroupId>
        <GroupVersion>00</GroupVersion>
        <GroupDescription>Group 2</GroupDescription>
        <version_date>199708101105</version_date>
        <superuser_flag>false</superuser_flag>
        <users>22 23</users>
    </UserGroup>
    <UserGroup>
        <GroupName>TZU 23</GroupName>
        <GroupId>50</GroupId>
        <GroupVersion>00</GroupVersion>
        <GroupDescription>Group 3</GroupDescription>
        <version_date>199708101105</version_date>
        <superuser_flag>false</superuser_flag>
        <users>21 23</users>
    </UserGroup>
<!--
****************************************************************************************************
*   Access Control List information for enforcing security
****************************************************************************************************
-->
    <ACLA>
        <users>30</users>
        <appl_element_id>333</appl_element_id>
        <rights>7</rights>
        <!--- protection of whole application element -->
        <attribute_name/>
    </ACLA>
    <ACLA>
        <users>30</users>
        <appl_element_id>333</appl_element_id>
        <rights>3</rights>
        <!--- protection of description attribute -->
        <attribute_name>description</attribute_name>
        </ACLA>
        <ACLA>
            <users>40</users>
            <appl_element_id>123</appl_element_id>
            <rights>10</rights>
            <!--- protection of whole application element -->
            <attribute_name/>
        </ACLA>
        <ACLI>
            <users>50</users>
            <appl_element_id>123</appl_element_id>
            <rights>10</rights>
            <!--- protection of instance element -->
            <instance_id>23</instance_id>
        </ACLI>
        <ACLI>
            <users>40</users>
            <appl_element_id>123</appl_element_id>
            <rights>10</rights>
            <!--- protection of instance element -->
            <instance_id>23</instance_id>
        </ACLI>
    </instance_data>
</atfx_file>
```

6.13 KNOWN PROBLEMS

At the time of this writing, the XML Include specification had not been finalized by the W3C and support for it is not yet universal.

At the time of this writing, the ASM XML Styleguide specification is under review and this document could be impacted by any changes to the Styleguide.

6.14 ADDITIONAL RESOURCES

For more information on XML, XML Schema, and XML Inclusions, please consult the W3C XML Documents.

Extensible Markup Language (XML) at http://www.w3.org/XML/

XML Schema Part 1: Structures at http://www.w3.org/TR/xmlschema-1/

XML Schema Part 2: Data Types at http://www.w3.org/TR/xmlschema-2/

XML Inclusions (XInclude) Version 1.0 at http://www.w3.org/TR/xinclude/

6.15 REVISION HISTORY

Date Editor	Changes
2003-06 M. Quinsland	Created document
2003-12-29 R. Bartz	Updated document format and references to the other parts of the ASAM ODS documentation Included some explanation on data type usage
2003-12-20 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 8
MIME TYPES & EXTERNAL REFERENCES

Version 1.0

Association for **S**tandardisation of
Automation and **M**easuring Systems

Dated: 30.09.2004

Status of Document

Reference:	ASAM ODS Version 5.0 MIME Types
Date:	30.09.2004
Author:	Hans-Peter Daunert, BMW
Type:	Specification
Doc-ID:	ASAM_ODS_50_CH08_Mime_Types.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 (odsftr@asam.net) to make sure this issue will be addressed within the next review cycle.

Contents

8 MIME TYPES AND EXTERNAL REFERENCES 8-3

8.1 INTRODUCTION 8-3
8.2 DEFINITION OF ASAM ODS MIME TYPES 8-4
8.2.1 MIMETYPE AS AN OPTIONAL BASE ATTRIBUTE 8-4
8.2.2 CONTENT OF A MIMETYPE STRING 8-4
8.2.3 READING MIME TYPES VIA THE ODS APIS 8-5
8.2.4 WRITING MIME TYPES VIA THE ODS APIS 8-5
8.3 ASAM ODS MIME TYPES 8-6
8.4 HANDLING EXTERNAL REFERENCES IN ASAM ODS 8-7
8.5 REVISION HISTORY 8-8

Scope

This document describes the MIME types of the ASAM ODS Version 5.0.

Intended Audience

This document is intended for implementers of ASAM ODS Version 5.0. It shall be used as a technical reference for MIME types used in 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

8 MIME TYPES AND EXTERNAL REFERENCES

8.1 INTRODUCTION

The standard ASAM ODS is used as data integration standard in the test environment.

Additionally to e.g. normal test results like measurements there will always be a lot of data which doesn't fit into the specified object types. Because of their references to application elements of type AoTest, or AoUnitUnderTest, or others it is necessary to support an integration of those data (e.g. a Fast Fourier Transformation analysis as type of an AoMeasurement inside an ODS environment or a video of a crash test of an AoUnitUnderTest outside the ODS environment).

The types of integrated data are manifold and must not be limited by ODS (video, sound, analysis data, CAD models, ….). Thus there has to be a generic way to include information about the data and allow to store them either within the ODS physical storage or to specify references to existing data outside the ODS data store.

To specify the information about the type of data stored, ASAM ODS uses the MIME type definition. This chapter defines how the concept of MIME type is applied to ASAM ODS. Additionally it defines how ASAM ODS supports the referencing to any external objects (outside ODS), using MIME types.

8.2 DEFINITION OF ASAM ODS MIME TYPES

8.2.1 MIMETYPE AS AN OPTIONAL BASE ATTRIBUTE

"MimeType" is a base attribute like e.g. "version". If the attribute exists, its value supports the rules listed below. The MimeType in ODS is only an additional information (attribute) of the application elements.

Using the defined MIME type mechanisms doesn't include the definition of special functionality (e.g. ODS-server can have some functionality, ATF file doesn't).

For ODS servers no additional functionality is necessary. All necessary functionality is needed in the applications reading and writing this attribute.

8.2.2 CONTENT OF A MIMETYPE STRING

The name of the attribute is "MimeType"

The type of the attribute is related to the data type enumeration name DT_STRING".

Every MimeType string has to be written in LOWERCASE.

The MimeType string is built from the following parts:

`<application/x-> <asam> "." <standardtype> "." (<subtype>)`

Meaning of the different elements:

<application/x->:	ASAM MIME types are in the group "application/x-". (after registration they will be in group "application")
<asam>:	The first part of the ASAM MimeType string. Defines that it's an ASAM Type
<standardtype>:	E.g. "aomeasurement". Defines what standardized MIME type it is (all ASAM ODS base elements are standard types, e.g. aoany, aouser).
(<subtype>):	the standardized MimeType string can have additional subtypes (e.g. "special-temperature-measurement" as subtype of AoMeasurement). Subtypes are typically defined within specific application models.

After the group ("application/x-" or after registration "application") the parts have to be separated by a Dot.

NOTE: All parts except of <subtype> are already known inside the ODS environment (e.g. everybody knows "aomeasurement" because of the base ID of AoMeasurement). To reduce redundancies ASAM ODS defines:

➢ Inside the ODS environment only the optional <subtype> has to be used as short form of the MimeType string (e.g. "special-temperature-measurement").

- Every application exporting or bridging ODS objects outside the ODS environment has to publish the long form MimeType string including all parts (e.g. export to ATF or bridging into ASAM-CEA bus structure publishes "application/x-asam.aomeasurement.special-temperature-measurement").

With these rules MimeType strings can be for example:

"*application/x*-asam.aomeasurement.special-temperature-measurement",

"*application/x*-asam.aotest.fatiguetest"

or the short form inside ODS environments:

"specialtemperature"

Every base element of ASAM ODS is also a standardized MIME type (e.g. "AoAny", "AoSubmatrix", "AoUser", etc. For a list of all base elements see the data model of ASAM ODS)

8.2.3 READING MIME TYPES VIA THE ODS APIS

Because the MimeType attribute is optional, at first every client has to look if the application element has the attribute "MimeType" (case insensitive).

If there is the attribute MimeType the client has to check whether it is filled in the Instance.

If it is available and filled the client can be sure that the string is a valid MimeType string following the rules listed below.

NOTE: Every application should be aware that there are two types of MimeType strings:

- Long MIME type: Includes the whole MimeType string. Used only outside the ASAM ODS environment
- Short MIME type: Only the Subtype without any ODS-standardized prefix. Used only inside the ODS environment

8.2.4 WRITING MIME TYPES VIA THE ODS APIS

Every application writing ODS data should support the MimeType attributes wherever they are existing in the data model.

The writing application has to guarantee that the written MimeType string will support the rules listed below.

NOTE: Every application should be aware that there are two types of MimeType strings:

- Long MIME type: Includes the whole MimeType string. Used only outside the ASAM ODS environment.
- Short MIME type: Only the subtype without any ODS-standardized prefix. Used only inside the ODS environment.

8.3 ASAM ODS MIME TYPES

The following table shows the MIME types for ASAM ODS Version 5.0:

ASAM ODS MIME types
application/x-asam.aoany
application/x-asam.aoattributemap
application/x-asam.aoenvironment
application/x-asam.aoexternalcomponent
application/x-asam.aolocalcolumn
application/x-asam.aolog
application/x-asam.aomeasurement
application/x-asam.aomeasurementquantity
application/x-asam.aonamemap
application/x-asam.aoparameter
application/x-asam.aoparameterset
application/x-asam.aophysicaldimension
application/x-asam.aoquantity
application/x-asam.aoquantitygroup
application/x-asam.aosubmatrix
application/x-asam.aosubtest
application/x-asam.aotest
application/x-asam.aotestdevice
application/x-asam.aotestequipment
application/x-asam.aotestequipmentpart
application/x-asam.aotestsequence
application/x-asam.aoteststep
application/x-asam.aounit
application/x-asam.aounitgroup
application/x-asam.aounitundertest
application/x-asam.aounitundertestpart
application/x-asam.aouser
application/x-asam.aousergroup

8.4 HANDLING EXTERNAL REFERENCES IN ASAM ODS

Referencing any kind of object outside the ODS environment is done by another optional Base Attribute, named “externalReference”. It is in the data model designer’s responsibility to decide which application elements will have the attribute “externalReference”.

The responsibilities for reading and writing are already described in sections 8.3.3 and 8.3.4.

This attribute will be used (optionally) on every application element of ODS. It contains a list [0..n] of 3 strings; the data type of each of them is specified by the DataType enumeration name DT_STRING:

description:

This is a textual description that may be used for storing some small description of the object, so that the client does not need to open the object to get first information (some “advance information”).
An example for this description is: "This object contains results of the full load test 123".

MimeType:

Used for storing the **MIME type** of the external object (rules for the **MimeType** string see above).

location:

Used for storing a clear locator string for accessing the object. It is based on the ideas of the URL and the CORBA 3.0 URL definitions and is composed of the following parts:

<protocol>://[<version>@][<host address>]/<key string>

(For further information to standard “URL” definitions have a look at the published specifications of “URL” in the internet).

ASAM ODS defines for ODS objects the following locator string:

`“asamods://[<version>@]<asampath>”`

The meaning of the respective elements is as follows:

<version>:	Version of the ODS standard. It contains • <major>: The released major number of the ODS standard • “.” • <minor>: The released minor number of the ODS standard A valid version would be “5.0”
<asampath>:	The ASAM path of the object.

ASAM ODS will support the references to any external object (files, servers, …). It is not intended to evaluate any of these objects in ODS servers. Responsible for using the referenced objects by reading, writing, viewing and so on are only the appropriate applications. ASAM ODS is only responsible for holding and exporting the reference to the objects via its API.

8.5 REVISION HISTORY

Date Editor	Changes
2003-10-13 R. Bartz	Several errors have been fixed.
2003-11 R. Bartz	Some details were removed that are not agreed upon within ASAM as a whole The MIME types have been limited to those describing base elements
2003-12-30 R. Bartz	The **Release** version has been created

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 9
RPC-API
VERSION 3.2.1

Association for **S**tandardization of
Automation and **M**easuring Systems

Dated: 30.09.2004

Status of Document

Reference:	ASAM ODS Version 5.0 RPC-API
Date:	30.09.2004
Author:	Gerald Sammer, AVL; Karst Schaap, HighQSoft
Type:	Specification
Doc-ID:	ASAM_ODS_50_CH09_RPC_API.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 (odsftr@asam.net) to make sure this issue will be addressed within the next review cycle.

Contents

9 THE ASAM ODS VERSION 5.0 RPC-API 9-5

9.1 PROCEDURE DECLARATIONS 9-5
9.2 DATA TYPES USED 9-5
9.3 SESSION SERVICES (OPEN, CLOSE) 9-7
9.3.1 AOP_OPENENV 9-7
9.3.2 AOP_CLOSEENV 9-9
9.4 META INFORMATION SERVICES 9-10
9.4.1 AOP_GETAPPLINF 9-10
9.4.2 AOP_GETATTR 9-11
9.5 THE APPLICATION ELEMENT VALUE SERVICES 9-12
9.5.1 AOP_GETINSTREF 9-12
9.5.2 AOP_SETINSTREF 9-13
9.5.3 AOP_GETVAL 9-14
9.5.4 AOP_GETVALE 9-17
9.5.5 AOP_GETINSTATTR 9-22
9.5.6 AOP_PUTVAL 9-23
9.6 MEASUREMENT AND PARTIAL MATRIX SERVICES 9-24
9.6.1 AOP_GETVALATTR 9-24
9.6.2 AOP_GETVALINF 9-26
9.6.3 AOP_GETVALVAL 9-27
9.6.4 AOP_PUTVALVAL 9-29
9.7 PROPERTY HANDLING 9-31
9.7.1 AOP_SETPAR 9-31
9.7.2 POSSIBLE PARAMETERS: 9-31
9.7.3 AOP_GETPAR 9-32
9.8 SECURITY SERVICE 9-33
9.8.1 AOP_SETSECURITYLEVEL 9-36
9.8.2 AOP_GETSECURITYLEVEL 9-37
9.8.3 AOP_SETRIGHTS 9-38
9.8.4 AOP_GETRIGHTS 9-38
9.8.5 AOP_SETINIRIGHTS 9-38
9.8.6 AOP_GETINIRIGHTS 9-39
9.8.7 AOP_SETINIRIGHTSREF 9-39
9.8.8 AOP_GETINIRIGHTSREF 9-40
9.8.9 AOP_SETCURRENTINIRIGHTS 9-40
9.8.10 AOP_SETPASSWORD 9-41
9.8.11 USER AUTHENTICATION 9-41
9.8.12 PASSWORD ENCRYPTION IN ODS VERSION 5.0 (RPC) 9-41
9.9 ERROR HANDLING 9-44

9.9.1 AOP_GETERR 9-45
9.10 RESTRICTIONS OF THE RPC-API..........9-46
9.11 EXAMPLE CALLING SEQUENCE 9-47
9.12 ASAM ODS VERSION 5.0 RPC-API IDL..........9-48
9.13 REVISION HISTORY 9-77

Scope

This document describes the ASAM ODS Version 5.0 RPC-API.

Intended Audience

This document is intended for implementers of ASAM ODS Version 5.0 (RPC-API). It shall be used as a reference on the RPC-API for 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

9 THE ASAM ODS VERSION 5.0 RPC-API

9.1 PROCEDURE DECLARATIONS

The procedures have numeric identifiers (program number) and version numbers. The version of the interface definition is communicated to get the proper procedure with proper arguments called. This allows running multiple versions concurrently. The RPC-API specified in version 5.0 of ASAM ODS is compatible with the former RPC-API version 3.2. To avoid recompilation of clients, the version in the interface-definition (aods.x) has not been changed.

9.2 DATA TYPES USED

ASAM has specified standardized data types which have been published end of 2002. These data types show standardized names, always beginning with A_. Those data types will be used by ASAM ODS consistently in all new specifications in the future.

The RPC-API has been developed and specified several years ago. At that time no standardized data types have been available ASAM-wide. Therefore ASAM ODS decided to define data types that cover the needs of ASAM ODS. These data types always start with T_.

The RPC-API has been implemented in ODS servers and installed in a quite large number of companies. Replacing the T_ types by the A_ types would require to modify and exchange those installed servers and also to modify the clients accessing the servers via the RPC-API (because the interface definition file has been changed).

Additionally, ASAM ODS does not intend to further extend or modify the RPC-API; its current specification is considered to be frozen.

That is why ASAM ODS has decided to not introduce the standardized ASAM data types (A_) in the RPC-API description and interface definition file.

Instead, chapter 2 of the ASAM ODS specification describes the relationship between the T_ types and the A_ types. One should note that the enumeration of the data types and their enumeration names are identical to those specified in the ASAM data type specification, and that the data types themselves are in most cases binary compatible. So a one-to-one mapping is easily possible, if needed.

Note: The base model requests the ID of an element to be a 64 bit integer value. Section 2.5.3 describes that a 32 bit integer may be used alternatively. This is how element IDs are represented here.

9.3 SESSION SERVICES (OPEN, CLOSE)

9.3.1 AOP_OPENENV

Open Environment

This has to be the first call to the server. The server will check access authorization and creates a session. Properties may be set using *AOP_SetPa*r (see below) and by using args parameters on *AOP_OpenEn*v. The environment Id returned by this call has to be used on all subsequent calls (the network address is saved as an additional identifier to be checked on subsequent calls). A redundant *AOP_OpenEnv* returns S_OK without handling properties again when a session is already established.

Every new session will get its new environment id, even on client connections from multi-user systems like UNIX.

```
OpenEnvRet AOP_OpenEnv(OpenEnvReq) = 1;
struct OpenEnvReq {            /* to open env. */
        AOP_NameV nvSeq<>;     /* environment open arguments */
};

union OpenEnvRet switch (AOP_Status retState) {

  case E_BSS: void;
  case E_SSS: void;
  case E_SECURITY: void;
  case E_MISC: void;
  case E_LICENSE: void;
  default: AOP_Id envId;       /* descriptor/handle/id whatever */
};
```

Note, that any default return state (e.g. E_SECURITY) indicates an error. E_BSS may indicate a resource problem, e.g. maximum number of processes or file descriptors exceeded.

EXAMPLE:

Example for nvSeq regarding security.
In this example the call to AOP_OpenEnv is missing.

```
/* Create rpc client. */
   cl = clnt_create (nodename, rpcNumber, AODSVERS3,"tcp");
   /* RPC Client creation failed ? */
   if (!cl) {
      /* Log connection failure. */
      sprintf (stderr, "Connect failed to %s\n", nodename);
      /* Handle client create error gracefully. */
      clnt_pcreateerror (nodename);
      /* Return with error status. */
      return (0);
   }
   /* Set timeout parameter. */
   if (clnt_control (cl, CLSET_TIMEOUT,(char *)&timeout) != TRUE){
      sprintf (stderr, "Unable to set timeout %s\n", nodename);
      return (0);
   }
   /* Create credential.
    * Since no keyserver is available on Linux, DES isn't possible.
    */
   cl->cl_auth = authunix_create_default ();
   /* Authentication valid ? */
   if (!cl->cl_auth) {
      sprintf (stderr, "No authentication to %s\n", nodename);
     return(0)
   }
```

9.3.2 AOP_CloseEnv

Close Environment

Close the environment, opened with OpenEnv. If the environment is opened more then once, the environment must be closed also more then once.

```
CloseEnvRet AOP_CloseEnv(CloseEnvReq) = 2;
```

```
struct CloseEnvReq  {     /* to close open env. */
        AOP_Id envId;
};

struct CloseEnvRet {
        AOP_Status retState;
};
```

9.4 META INFORMATION SERVICES

Get the application structure and information how an application structure maps to the basic structure.

9.4.1 AOP_GETAPPLINF

Get application structure information.

This function returns the whole information about the application structure. It is a list of application elements (Identifier = AID) together with application name and type (basic element AOP_BasElem) and a list of references between application elements.

```
GetApplInfRet AOP_GetApplInf(GetApplInfReq) = 14;
struct GetApplInfReq {
  AOP_Id        envId;            /* environ. handle */
};

struct ApplInfSeq {               /* application element infos */
  AOP_Id        aiAId;            /* application element id */
  AOP_BasElem   aiBId;            /* basic element Id */
  AOP_Name      aiName;           /* application element name */
};

struct ApplRelSeq {               /* application references infos */
  AOP_Id        arAId1;           /* from (0,1) resp. (n) */
  AOP_Id        arAId2;           /* to   (n)   resp. (m) */
  long          arRefNr;          /* attr number, 0 for n:m */
  AOP_Name      arName;           /* reference name */
  AOP_Flags     arConstr;         /* Constraint (e.g. not NULL) */
};

struct ApplInf {
  ApplInfSeq    aiSeq<>;          /* sequence of appelem info */
  ApplRelSeq    arSeq<>;          /* sequence of appref info */
};

union GetApplInfRet switch (AOP_Status retState) {
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: ApplInf  applInf;
};
```

9.4.2 AOP_GETATTR

Get Attribute Information of an application element

The client specifies an element-Id (aid and iid) in the request. If the instance-id (part of the element-Id) is *Any* (=0) only application attributes will be returned, otherwise attributes of the addressed instance will be included.

```
GetAttrRet AOP_GetAttr(GetAttrReq) = 12;
```

```
struct AttrSeq {              /* sequence of attributes */
  AOP_Name      aBName;       /* basic name */
                              /* 0 if not basic */
  AOP_Name      aAName;       /* appl. name */
  AOP_DataType  aDataType;    /* attribute data type */
  AOP_Id        aUnit;        /* unit if global defined */
};


struct AttrInf {              /* attribute info including header */
  AOP_BasElem   aBId;         /* basic element Id */
  AOP_Id        aAId;         /* application element Id */
  AttrSeq       aSeq<>;
};

struct GetAttrReq {           /* request for attribute list */
  AOP_Id        envId;        /* environ. handle */
  AOP_ElemId    elemId;       /* application element */
};

union GetAttrRet switch (AOP_Status retState) {
                              /* return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AttrInf attrInf;    /* including entity */
};
```

There are currently no methods provided to change the metainformation through the protocol layer, hence using the Version 5.0 published services, the meta-information is readonly.

For compatibility reasons, the structures should be left as in ODS 3.0.

This call is only used for returning the application structure, no instance-information. It returns the names of the attributes only.

For instance attributes use AOP_GetInstAttr. This function returns the names of the Instance-Attributes, which are attributes that are not part of the application model.

9.5 THE APPLICATION ELEMENT VALUE SERVICES

Includes fetching and storing application element instances using application element references.

9.5.1 AOP_GETINSTREF

Get Instance References

This query returns all instances from an application element referenced by the specified instance. Which application elements are referenced will be returned by AOP_GetApplInf. With the reference name multiple application references between 2 application elements can be distinguished. If an empty string is provided the base reference (which is always unique according to basic structure definition) is used. A given string is matched against the available reference names. On n:m references the name is taken from SVCREF otherwise from SVCATTR.

```
GetInstRefRet AOP_GetInstRef(GetInstRefReq) = 25;
```

```
struct GetInstRefReq {
  AOP_Id        envId;            /* environ. handle */
  AOP_ElemId    elemId;           /* instance */
  AOP_Name      refName;
  AOP_Id        aId;              /* of type */
};


union GetInstRefRet switch (AOP_Status retState) {
                                  /* return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_ElemId elemList<>;
};
```

9.5.2 AOP_SETINSTREF

Set Instance References

This method is used to create or remove n:m references.

AOP_PutVal is used for the n:1 case.

```
SetInstRefRet AOP_SetInstRef(SetInstRefReq) = 28;
```

```
struct SetInstRefReq {
  AOP_Id        envId;    /* environ. handle */
  AOP_ElemId    elemId1;  /* one element */
  AOP_ElemId    elemId2;  /* the other */
  AOP_Name      refName;
  long          onoff;    /* insert (onoff=1) or remove(onoff=0)*/
};

struct SetInstRefRet {
   AOP_Status retState;
};
```

9.5.3 AOP_GETVAL

Get Values of Application Elements

This is the standard method for retrieving values. Two different methods are implemented:

- By reference
  Used mainly in browsing (e.g. *give me all measurements belonging to test with Id 723* or *give me the testinstance belonging to (being parent from) measurement with Id 789*5). This follows the **”has a”** relationship within the AODS data model in both directions as described in API documentation as *Instance reference*s.
- By value
  Used when selecting instances from an application element by specifying conditions to be satisfied by the selection result. Since the AODS protocol doesn’t include yet a query language only limited possibilities exist, e.g.

  ```
  select * from test where date='19951201000000' and testtype='Endurance Test'.
  ```

  Selection of method is done by specifying either the reference elementId or by supplying a nonempty select list. If both are given selection by reference is done first, then the result is checked for select list match (Using RDBMS both methods are done in one step). elemId and refName are treated the same way as on AOP_GetInstRef (see above).

```
GetValRet AOP_GetVal(GetValReq) = 21;
```

```
struct AOP_NameS {          /* by value select */
  AOP_Name      name;       /* attribute name */
  AOP_ValMap    valMap;     /* range or pattern */
  AOP_SelOpcode selOpcode;  /* type of selection */
};
typedef AOP_NameS NSSeq<>;

struct GetValReq {            /* selection request structure */
  AOP_Id        envId;        /* environ. handle */
  AOP_Id        applId;       /* application element Id (from) */
  AOP_NameU     nuSeq<>;      /* sequence of to be reported attr's */
  AOP_NameS     nsSeq<>;      /* sequence of select fields */
  AOP_ElemId    elemId;       /* child/parent reference (where) */
  AOP_Name      refName;      /* named reference identifier */
};

union GetValRet switch (AOP_Status retState) {
                              /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_NameV nvSeq<>; /* sequence of resulting attributes */
};

enum AOP_SelOpcode {           /* Type of selection */
        SOMATCH,               /*     match                 ==      */
        SOUMATCH,              /*     unmatch               !=      */
        SORANGE,               /*     range                 < x >   */
        SOLESS,                /*     less                  <       */
        SOGREATER,             /*     greater               >       */
        SOLESSEQ,              /*     less equal            <=      */
        SOGREATEREQ,           /*     greater equal         >=      */
        SOINSET,               /*     within set            in      */
        SONOTINSET,            /*     not within set        !in     */
        SOORDER,               /*     order by                      */
        SOGROUP,               /*     group by                      */
        SOINSENSITIVE,         /*     insensitive search            */
        SOLIKE,                /*     like                  GetValE */
        SONOTLIKE,             /*     not like              GetValE */
        SONULL,                /*     is NULL               GetValE */
        SONOTNULL,             /*     is NOT NULL           GetValE */
        SOOPAND,               /*     operator AND          GetValE */
        SOOPOR,                /*     operator OR           GetValE */
        SOOPNOT,               /*     operator NOT          GetValE */
        SOOPBOPEN,             /*     operator bracket open ( GetValE */
        SOOPBCLOSE             /*     operator bracket close )GetValE */
};
```

Note that the `AOP_NameS.valMap` structure may contain one (SOMATCH, ...) or two (SORANGE) or 1 to n values (SOINSET, SONOTINSET). To cover the AODS requirement of making data access by explicitly specifying instance id (direct access to single element) it is necessary to set AOP NameS to

```
name = <IdAttr>
valMap.AOP_ValMap_u.lval.lval[0] = <iid>
selOpcode = SOMATCH
```

where IdAttr is the ID - basic attribute of an application element reported using GetAttr but with application name given. To get the ID-Application attribute name one may e.g. use assuming alret pointing to the structure returned by aop getattr:

```
char* GetIdAttr(GetAttrRet *alret) {
AttrSeq *aSeq;                          /* attribute info */
int aNr;                                /* attribute number */
aSeq = alret->GetAttrRet_u.attrInf.aSeq.aSeq_val;
  for (aNr = 0; aNr < alret->GetAttrRet_u.attrInf.aSeq.aSeq_len; aNr++) {
if (!stricmp(aSeq->aBName,"ID")
return aSeq->aAName;                    /* pointer to application name */
aSeq++;
  }
return NULL;
}
```

Please note:

- In SOMATCH no wildcards are allowed for strings. Use SOLIKE or SONOTLIKE instead.
- SOINSENSITIVE is a case insensitive comparison of strings.
- SOORDER and SOGROUP do not need a valMap in the nsSeq, because these operators only hold attributes, no values.

9.5.4 AOP_GETVALE

Get Values of Application Elements (extended Version for ODS Version 5.0 (RPC))

This method should be used, if one needs to select data from more than one application element or wants to use boolean operators. A segmented request is possible.

```
GetValERet AOP_GetValE(GetValEReq) = 26;
```

```
/*Enhanced Query structure, allows
   - joins of application elements
   - to specify ranges and patterns (Values are always arrays.
         This allows i.e. on selection to pass 2 or
         more values for e.g.: in/notin, range from - to)
   - to group selections with AND, OR and brackets (open, close)
   - to sort the result (order by)
   - to specify wildcards (including escape character)
   - partial access of the result (e.g. first 100, next 100, ..)

   To cover the AODS requirement of making data access by explicitly
   specifying instance id (direct access to single element) it is
   necessary to define the selection elements AOP_AIDNameS with
      selOpcode = SOMATCH
        AID = <application element ID>
        name = <Idattr>
        value.AOP_ValMap_u.lval.lval[0] = <iid>
   IdAttr is the first aBName reportet using GetAttr
*/
enum AOP_SelOpcode {             /* Type of selection */
        SOMATCH,                 /*     match           ==      */
        SOUMATCH,                /*     unmatch         !=      */
        SORANGE,                 /*     range           < x >   */
        SOLESS,                  /*     less            <       */
        SOGREATER,               /*     greater         >       */
        SOLESSEQ,                /*     less equal      <=      */
        SOGREATEREQ,             /*     greater equal   >=      */
        SOINSET,                 /*     within set      in      */
        SONOTINSET,              /*     not within set  !in     */
        SOORDER,                 /*     order by                */
        SOGROUP,                 /*     group by                */
        SOINSENSITIVE,           /*     insensitive search      */
        SOLIKE,                  /*     like                    */
        SONOTLIKE,               /*     not like                */
        SONULL,                  /*     is NULL                 */
        SONOTNULL,               /*     is NOT NULL             */
        SOOPAND,                 /*     operator AND            */
        SOOPOR,                  /*     operator OR             */
        SOOPNOT,                 /*     operator NOT            */
        SOOPBOPEN,               /*     operator bracket open ( */
        SOOPBCLOSE               /*     operator bracket close )*/
};

/* ------------------------- AID + Name + Unit */
struct AOP_AIDNameU {
  AOP_Id        aid;             /* application element ID */
  AOP_Name      name;            /* attribute name */
  AOP_Id        unitId;          /* requested unit */
};
```

```
typedef AOP_AIDNameU ANUSeq<>;

/* -------------------------- AID + Name + Unit + Value(s) */
struct AOP_AIDNameV {
  AOP_Id        aid;              /* application element ID */
  AOP_Name      name;             /* attribute name */
  AOP_Id        unitId;           /* requested unit */
  AOP_ValMap    valMap;           /* values */
};
typedef AOP_AIDNameV ANVSeq<>;

enum AOP_ReportListType {         /* type of report list */
  RLCOMPLETE,                     /* all attributes */
  RLSELECTIVE                     /* given list of attributes */
};
union AOP_ReportList switch (AOP_ReportListType rlType) {
  case RLCOMPLETE:
    AOP_Id       aidSeq<>;        /* list of application elements - all
                                     attributes will be reported */
  case RLSELECTIVE:
    AOP_AIDNameU anuSeq<>;        /* list of attribute to be reported */
  default: void;
};
struct AOP_AIDNameS {             /* by value select */
  AOP_SelOpcode selOpcode;        /* type of selection */
  AOP_Id        aid;              /* application element ID */
  AOP_Name      name;             /* attribute name */
  AOP_ValMap    valMap;           /* range or pattern */
  AOP_SortDir   dir;              /* sort order */
};
typedef AOP_AIDNameS ANSSeq<>;
struct AOP_RefDef {               /* reference definition */
  AOP_Id        fromAid;          /* start point (AID) of reference */
  AOP_ElemId    elemId;           /* child/parent reference (where) */
  AOP_Name      refName;          /* named reference identifier */
};

enum AOP_ReqType {                /* type of request */
  RTDEFAULT,                      /* force query and close cursor */
  RTOPEN,                         /* force query and keep cursor open */
  RTCONTINUE,                     /* continue reading cursor */
  RTCLOSE                         /* close open cursor */
};

struct GetValEReq {               /* enhanced query request structure */
  AOP_Id         envId;           /* environ. handle */
  AOP_ReqType    reqType;         /* request type */
  AOP_ReportList repList;         /* definition of report list */
  AOP_AIDNameS   nsSeq<>;         /* sequence of conditions */
  AOP_RefDef     refDef;          /* reference definition */
  long           rowCnt;          /* number of requested rows
                                     - rowCnt<=0 report all rows */
};

union GetValERet switch (AOP_Status retState) {
                                  /* select return value structure */
  case E_NOTOPEN: void;
```

```
    case E_BSS: void;
    case E_SSS: void;
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: ANVSeq result<>;     /* sequence of resulting attribute
                                        sequences */
  };
```

The application elements in the AOP_ReportList must have a reference from or to the application element used by the AOP_RefDef.fromAid or the application element used in the attribute value selection AOP_AIDNameS.aid. If there is no reference the instances of the application element will not be reported.

EXAMPLE:

This example demonstrates the use of the aop_getvale function. A new request is set up with the characteristics

- no segmentation
- no attributes defined - only application element will be defined
- no select criteria
- no joins explicitly defined - not necessary if not ambiguous. This example does not use joins.
- no referenced instance given

The last line contains the method invocation.

```
GetValEReq vereq;
GetValERet *veret;
…
vereq.envId = sessid;
vereq.reqType = RTOPEN; /* force a new select */
vereq.rowCnt = -1; /* report the whole result */

/* request all attributes (select *) */
vereq.repList.rlType = RLCOMPLETE; /* select * */
vereq.repList.AOP_ReportList_u.aidSeq.aidSeq_len = 1;
vereq.repList.AOP_ReportList_u.aidSeq.aidSeq_val =
malloc(sizeof(AOP_Id));
vereq.repList.AOP_ReportList_u.aidSeq.aidSeq_val[0] = aid;

/* no select conditions */
vereq.nsSeq.nsSeq_len = 0;
vereq.nsSeq.nsSeq_val = NULL;

/* no explicit join definition */
vereq.jdSeq.jdSeq_len = 0;
vereq.jdSeq.jdSeq_val = NULL;

/* no reference definition used */
vereq.refDef.fromAid = 0;
vereq.refDef.refName = strdup("");
vereq.refDef.elemId.aid = 0;
vereq.refDef.elemId.iid = 0;

veret = (GetValERet *) aop_getvale_3 (&vereq, cl);
```

EXAMPLE:

This code example relies on the following specifications:

- testseries-name is "MOTID"
- measurement-name is "Messung"
- quantity-name is "GroNameDef"
- quantity-group-name is "GroGruName"
- there is one select criteria: measurement-name = "267_FU"

```
/* request specific attributes */
vereq.repList.rlType = RLSELECTIVE; /* select with attribute list */
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_len = 2;
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val =
      malloc(2*sizeof(AOP_AIDNameU));
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[0].aid = 2;
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[0].name =
      strdup("MOTID");
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[0].unitId = 0;
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[1].aid = 4;
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[1].name =
      strdup("Messung");
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[1].unitId = 0;


/* one select condition */
vereq.nsSeq.nsSeq_len = 1;
vereq.nsSeq.nsSeq_val = malloc(sizeof(AOP_AIDNameS));
vereq.nsSeq.nsSeq_val[0].selOpcode = SOMATCH;
vereq.nsSeq.nsSeq_val[0].aid = 4;
vereq.nsSeq.nsSeq_val[0].name = strdup("Messung");
vereq.nsSeq.nsSeq_val[0].valMap.dtyp = DT_STRING;
vereq.nsSeq.nsSeq_val[0].valMap.AOP_ValMap_u.aval.aval_len=strlen("2
      67_FU")+1;
vereq.nsSeq.nsSeq_val[0].valMap.AOP_ValMap_u.aval.aval_val=strdup("2
      67_FU");
vereq.nsSeq.nsSeq_val[0].dir = 0;

/* request specific attributes */
vereq.repList.rlType = RLSELECTIVE; /* select with attribute list */
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_len = 2;
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val =
      malloc(2*sizeof(AOP_AIDNameU));
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[0].aid = 7;
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[0].name=strdup("Gro
      NameDef");

vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[0].unitId = 0;
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[1].aid = 11;
vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[1].name=strdup("Gro
      GruName");

vereq.repList.AOP_ReportList_u.anuSeq.anuSeq_val[1].unitId = 0;
```

EXAMPLE:

In this example code there is also one select criteria:
quantity-group-name = "B*"

```
/* one select condition */
vereq.nsSeq.nsSeq_len = 1;
vereq.nsSeq.nsSeq_val = malloc(sizeof(AOP_AIDNameS));
vereq.nsSeq.nsSeq_val[0].selOpcode = SOLIKE;
vereq.nsSeq.nsSeq_val[0].aid = 7;
vereq.nsSeq.nsSeq_val[0].name = strdup("GroNameDef");
vereq.nsSeq.nsSeq_val[0].valMap.dtyp = DT_STRING;
vereq.nsSeq.nsSeq_val[0].valMap.AOP_ValMap_u.aval.aval_len =
      strlen("B*") + 1;
vereq.nsSeq.nsSeq_val[0].valMap.AOP_ValMap_u.aval.aval_val =
      strdup("B*");
vereq.nsSeq.nsSeq_val[0].dir = 0;
```

EXAMPLE:

```
/* one join definition */
vereq.jdSeq.jdSeq_len = 1;
vereq.jdSeq.jdSeq_val = malloc(sizeof(AOP_JoinDef));
vereq.jdSeq.jdSeq_val[0].fromAid = 7;
vereq.jdSeq.jdSeq_val[0].toAid = 11;
vereq.jdSeq.jdSeq_val[0].refName = strdup("GROEBEZ");
vereq.jdSeq.jdSeq_val[0].joinType = JTOUTER;
```

9.5.5 AOP_GETINSTATTR

Get instance-attributes of some element.

Instance attributes are attributes that are not modeled in the physical (meta) data model. They can be attached to each instance of an element. Instance attributes are stored in a separate table and the attribute-name(s) and –value(s) can be retrieved with this method. The result depends on the values of elemId. If only elemId.aid is set, AOP_GetInstAttr will return information (name, data type and unit but not the values) about all Instance attributes available for the specified application element. If elemId referencing one instance (elemId.aid and elemId.iid are valid) all Instance attributes belonging to the specified instance will be returned. The data types DT_BLOB and DT_BYTESTR are not supported.

```
GetInstAttrRet          AOP_GetInstAttr(GetInstAttrReq) = 22;
```

```
struct GetInstAttrReq {        /* selection request structure */
  AOP_Id       envId;          /* environ. handle */
  AOP_ElemId   elemId;         /* application element Id */
};

union GetInstAttrRet switch (AOP_Status retState) {
                               /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_NameV nvSeq<>; /* sequence of resulting attributes */
};
```

9.5.6 AOP_PUTVAL

Store Operation of an Application Element

This request is used by the client to perform insert, update and delete operations on an application element. The request handles only one instance per call. Which operation the client requests, is implicitly given by the values in the request-structure. If there is no instance (iid == 0 in elemId) specified an insert operation is performed creating a new application element instance. The new instance-Id is returned. If the instance-id is specified, the operation depends on the name `nvSeq` argument. An empty `nvSeq` performs a delete operation on the given Id otherwise an update operation will be performed.

NOTE: The behavior on reimporting an instance element deleted before is not part of ODS. The server cannot distinguish between inserting a new instance and reimporting a deleted instance. Therefore, a reimported instance will get a new instance-id (addressing will be done via the ASAM path, anyway).

```
PutValRet   AOP_PutVal(PutValReq) = 27;
struct PutValReq {            /* update request structure */
  AOP_Id        envId;        /* environ. handle */
  AOP_ElemId    elemId;       /* application element Id */
  AOP_NameV     nvSeq<>;      /* sequence of values */
};

union PutValRet switch (AOP_Status retState) {
                              /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Id iid;        /* storage instance id */
};
```

9.6 MEASUREMENT AND PARTIAL MATRIX SERVICES

The following functions refer to the fixed (not meta-modeled) part of the implementation. All methods need the measurement application element instance Id as an input argument.

If no application element Id is given the server expects the application element of the type AoMeasurement.

9.6.1 AOP_GETVALATTR

Get Attribute Information of a Partial Matrix

The server returns the list of channels of a specified partial matrix and the number of measurementpoints in the partial matrix. The segmentation of the partial matrix is transparent to the client. If `pmatid` is 0 the information is returned for all partial matrices belonging to the given measurement.

```
GetValAttrRet AOP_GetValAttr(GetValAttrReq)
struct ValAttrSeq {           /* linked list of attributes */
  AOP_Name      name;         /* appl. structure name */
  AOP_Id        qtyId;        /* reference to dict. (0 allowed) */
  AOP_Id        unitId;       /* UnitId */
  AOP_IndepFl   indepFl;      /* independent flag */
  AOP_ImplFl    implFl;       /* implicit flag */
  AOP_DataType  dataType;     /* attribute data type */
};

struct ValAttr {              /* measurement attribute cursor list */
  AOP_Id        pmatId;       /* PMatId */
  long          numPnt;       /* number of measurement point
                                 in partial matrix */
                              /* should be extended for min/max */
  ValAttrSeq    vaSeq<>;      /* attribute sequence  */
};

struct GetValAttrReq {        /* request for attribute list */
  AOP_Id        envId;        /* environ. handle */
  AOP_ElemId    elemId;       /* measurement id */
  AOP_Id        pmatId;       /* partial matrix id (0 = all)*/
};

union GetValAttrRet switch (AOP_Status retState) {
                              /* return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: ValAttr valAttr<>;/* column info for 1 or more PMat's */
};
```

QtyId refers to an instance element (iid) of the application element with the type AoQuantity.

UnitId refers to an instance element (iid) of the application element with the base type AoUnit.

IndepFl = 1 means that the column is an independent column.

The implicit column indicator changes the interpretation of the values them self. Usually values are stored explicit. For some kind of data (e.g. timestamps on time driven sampling) the value for each row may be calculated. Following variants shall be provided (n = partial matrix row number):

value count (numPnt)	value in row n meaning
1	xn = x1 constant
2	xn = x1 + (n -1) * x2 linear expression
3	xn = x1 + ((n - 1) mod (x3 - x1)/x2) * x2 saw curve

Only numerical data types are allowed for implicit storage. The calculation is done using the data type defined for the column. The expression (x3 - x1)/x2 is truncated to integer to start each saw curve cycle at x1.

DT_DATE is not a numerical data type.

The data type DT_BLOB is not supported in measured values.

9.6.2 AOP_GETVALINF

Get Information of Partial Matrix

This function returns information about the partial matrices in a measurement.

It is similar to AOP_GetValAttr but the main purpose is to provide information on how multiple partial matrices are related using local column information.

```
GetValInfRet AOP_GetValInf(GetValInfReq) = 35;
```

```
struct ValInfSeq {              /* partial matrix interrelations */
  AOP_Id       pmatId;
  AOP_Name     pmatName;          /* sprintf ofpmatId */
  AOP_Name     presortNames <>;  /* columns to sort in pmat */
  AOP_SortDir  presortDir;        /* sortdirection */
  AOP_Name     joinNames <>;      /* join columns */
  AOP_JoinTyp  joinTyp;
  AOP_Name     postsortNames <>; /* columns to sort on result */
  AOP_SortDir  postsortTyp;
};

struct GetValInfReq {
  AOP_Id       envId;
  AOP_ElemId   elemId;            /* aid has to be measurement */
};

union GetValInfRet switch (AOP_Status retState) {
                                  /* info fetch return value
                                     structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: ValInfSeq iSeq<>;      /* sequence of info's */
};
```

The fields joinNames and joinTyp shows the relation between the different submatrices of the measurement. For the moment only the joinTyp MERGE is used and the joinNames are the names of the independent columns. The fields presortNames, presortDir, postsortNames and postsortDir are not used for the moment.

9.6.3 AOP_GETVALVAL

Get Values from a Measurement

This service returns measured data of one or more partial matrices. It is possible to address subsets by specifying columns and local measurement point numbers.

```
GetValValRet AOP_GetValVal(GetValValReq) = 32;
struct ValValSeq {             /* linked list of reported vectors */
  AOP_Name      name;          /*  MQ Name */
  AOP_Id        qtyId;         /* dictionary Id (ref) */
  AOP_Id        unitId;        /* unit Id */
  AOP_IndepFl   indepFl;       /* independent flag */
  AOP_ImplFl    implFl;        /* implicit (versus explicit) */
  AOP_ValMap    valMap;        /* vector for values */
  AOP_ValFlags  valFlags;      /* vector for values flags */
};

/* the partial matrix now holds 2 numbers to be able to preserve
 * origin of data
 */
struct ValVal {                /* local column values */
  AOP_Id        pmatId;        /* pMatId (0 .. yet undef) */
  long          mPntBase;      /* local point number base */
  long          mPntCnt;       /* # local points  */
  ValValSeq     vvSeq<>;       /* measurement value list */
};

struct GetValValReq {          /* request for measurement results */
  AOP_Id        envId;         /* environ. handle */
  AOP_ElemId    elemId;        /* elementId, bid to elemId.aid has
                                  to be BE_MEA */
  AOP_Id        pmatId;        /* partial matrix number */
  long          mPntBase;      /* from local measurement point
                                  number */
  long          mPntCnt;       /* number of measurement points
                                  requested */
  AOP_NameU     nuSeq<>;       /* sequence of attributes to report */
};

union GetValValRet switch (AOP_Status retState) {
                               /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: ValVal valVal<>;   /* partial matrices */
};
```

For better efficiency, all flags of one value are encoded in one byte, which means, for each flag one bit will be used. For access to the single flags, constants for the bit masks are defined. The following value flags are supported:

- AO_VF_VALID (0x01) - the value is valid

- AO_VF_VISIBLE (0x02) - the value has to be visualized
- AO_VF_UNMODIFIED (0x04) - the value has not been modified
- AO_VF_DEFINED (0x08) - the value is defined (this flag is also used by the base layer to mark gaps in the value matrix)

9.6.4 AOP_PutValVal

Put Values into a Measurement

To distinguish between update, insert, append and delete the information provided in `valVal` is interpreted as follows.

pmatId	mPntBase	mPntCnt	vvSeq len	action
0	any	any	0	invalid
0	0	0	>0	1.)
>0	0	0	0	2.)
>0	>0	0	0	invalid
>0	>0	>0	0	3.)
>0	0	any	>0	4.)
>0	>0	0	>0	5.)
>0	>0	>0	>0	6.)

1.) Append to the first matching sub matrix or if no matching sub matrix can be found create a new sub matrix with the specified number of local columns. The properties and values of the local columns are given in vvSeq_val.

2.) Delete whole sub matrix identified by pmatId

3.) Delete from all local columns of the specified sub matrix the number of elements (mPntCnt) beginning with the first specified element (mPntBase).

4.) Append to all local columns of the specified sub matrix the elements given in vvSeq_val. The number of local columns of the specified sub matrix must be identical with the number of local columns given in the request structure. The order of the local columns isn't significant, the name will be taken as the key. The local column description information such as implicit and independent flags has to be identical with the data in the server. If the sub matrix does not exist the sub matrix will be created. If there is a new local column, the client must not specify a pmatid (pmatid=0, see case 1.), because the server has to choose the right one or creates a new submatrix).

5.) Insert to all local columns of the specified sub matrix the elements given in vvSeq_val. The data will be insert after the specified first point. If the specified first point is outside the current size of the local column the data will be appended. The number of local columns of the specified sub matrix must be identical with the number of local columns given in the request structure. The order of the local columns isn't significant, the name will be taken as the key. The local column description information such as implicit and independent flags has to be identical with the data in the server.

6.) Replace the number of elements in the specified local columns of the specified sub matrix. The number of values of each local column (vvSeq_val.valMap.AOP_ValMap_u.lval.lval_len) must be identical with the specified number of elements (mPntCnt).

The last element to modify must be within the existing size of the local column. When the insert position exceeds the current size insert is automatically changed to append. Appending and inserting into an existing partial matrix is only allowed if all columns are supplied in one call (structure match, order of columns is not significant). Replacing is possible on a "per column" way but the number of points to be replaced must match the number of points provided in the call in this case. Currently there is no "*by value"* positioning possible, e.g. using the implicit columns to define the local measurement point numbers.

```
PutValValRet PutValVal(PutValValReq)
```

```
struct PutValValReq {        /* request for measurement results */
  AOP_Id        envId;       /* environ. handle */
  AOP_ElemId    elemId;      /* elementId, bid to elemId.aid has
                                to be BE_MEA */
  ValVal        valVal<>;    /* sequence of headers + columns */
                             /* (only one PMat for now per call) */
};

union PutValValRet switch (AOP_Status retState) {
                             /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Id pmatId;    /* pmatid given by server */
};
```

9.7 PROPERTY HANDLING

Properties are name-value pairs where the name is of type aods_Name and value is of type aop_ValMap to take any AODS-allowed data type including arrays. Such tuples can be passed to the server. They are not persistent. Main purpose is to parametrize the server behavior for one client (properties belong to the session). If there are other plugins at the server side (e.g. a formula interpreter) parameters can be used to provide additional information (e.g. selection between different kinds of formulas or supplementary variables used within the formulas).

9.7.1 AOP_SETPAR

Set Parameters

```
SetParRet AOP_SetPar(SetParReq) = 51;
struct SetParReq {
  AOP_Id        envId;                  /* environ. handle */
  AOP_NameV     nvSeq<>;                /* parameters */
};

struct SetParRet {
   AOP_Status retState;
};
```

9.7.2 POSSIBLE PARAMETERS:

Parameter	Used for	Example

USER	Set Username and Password	nvSeq[0].name = 'USER'
		nvSeq[0].valMap.aval = <Username>\0<Password>\0

WILDCARD_ONE	wildcard-character for one character match	nvSeq[0].name = 'WILDCARD_ONE'
		nvSeq[0].valMap.aval = "?"

WILDCARD_ALL	wildcard-character for zero or more characters match	nvSeq[0].name = 'WILDCARD_ALL'
		nvSeq[0].valMap.aval = "*"

WILDCARD_ESC	escape-character	nvSeq[0].name = 'WILDCARD_ESC'
		nvSeq[0].valMap.aval = "\\"

9.7.3 AOP_GETPAR

Get Parameters

```
GetParRet AOP_GetPar(GetParReq) = 52;
struct GetParReq {
  AOP_Id        envId;                  /* environ. handle */
  AOP_NameU     nuSeq<>;                /* parameter name-unit
tuples */
};

union GetParRet switch (AOP_Status retState) {
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_NameV      nvSeq<>;      /* parameters */
};
```

Parameters that can be queried:

Parameter	Value
USER	Username
ODSVERSION	supported Interface ODS-API Version (e.g. 5.0)
WILDCARD_ONE	wildcard-character for one character match
WILDCARD_ALL	wildcard-character for zero or more characters match
WILDCARD_ESC	escape-character

9.8 SECURITY SERVICE

In ODS Version 5.0 a new security handling has been introduced to support users, usergroups and different access-rights on elements, attributes and/or instances of elements. For a detailed description of the ODS Security see chapter 2.9.

Users and usergroups are application elements which can be accessed by standard API methods (AOP_Getval, AOP_Putval). Each user can be assigned to one ore more usergroups. These m:n references can be set with AOP_SetInstRef. All other services are exposed only by the Security-API-methods described below.

One has to distinguish between access-rights and access-right-templates. Access-rights are stored in the ACL (access control list). Access-right-templates are stored in IRL (initial rights list). An IRL is used to store "default-rights" that will be copied to the ACL of new instances.

```
typedef long AOP_Rights; /* Used for Rights-Values (bitmasked) */

enum AOP_SetType {        /* how to modify bitmasked values */
  SET,                    /* Set the value to new bits (clears all
                             bits before) */
  ADD,                    /* Adds the new bits to the existing ones
                            (OR-function) */
  REMOVE                  /* Removes the bits from the existing
                             value (~AND function) */
};

struct AOP_Acs {          /* Access control structure */
  AOP_Id     groupId;     /* Usergroup Id */
  AOP_ElemId elemId;      /* Application element+instance Id */
  AOP_Name   attrName;    /* Application attribute name */
  AOP_Rights rights;      /* rights-value (bitmasked) 5-bits */
};
struct AOP_Irs {          /* Initial rights structure */
  AOP_Id     groupId;     /* Usergroup Id */
  AOP_ElemId elemId;      /* Application element+instance Id */
  AOP_Rights rights;      /* rights-value (bitmasked) */
  AOP_Id     refAid;      /* referencing application element. If
                             set to 0, its been used by all
                             elements that refer to it */
};
```

The following five basic rights are specified:

Read	Update	Insert	Delete	Grant

Read:

- ➢ R-right on application element: The client may read all instances of this application element.
- ➢ R-right on instance: The client may read this instance.
- ➢ R-right on attribute: The client may read this attribute of all instances of the application element.

Update:
- U-right on application element: The client may modify all instances of this application element.
- U-right on instance: The client may modify this instance.
- U-right on attribute: The client may modify this attribute of all instances of the application element.

Insert:
- I-right on application element: The client may create new instances of this application element.
- I-right on instance: The client may create child instances for this (parent) instance.
- I-right on attribute: This does not make sense and will be ignored by the server (should not be supported by administration tool).

Delete:
- D-right on application element: The client may delete any instance of this application element.
- D-right on instance: The client may delete the specific instance.
- D-right on attribute: This does not make sense and will be ignored by the server (should not be supported by administration tool).

Grant:
- G-right on application element, instance or attribute: Access rights may be passed on. (This right is interpreted by the server as follows: If a user group has GRANT right on a data object, any of its members may pass the group's rights on the data object to other user groups).

Note (insert right on instance): Creating a child-instance means creating a new instance or reassigning an existing child to a different parent.

Note (insert right on instance): An I-right on an instance for an application element which is not a parent in any parent-child relationship does not make sense and will be ignored by the server (should not be supported by administration tool)

Read	Update	Insert	Delete	Meaning:
0	0	0	0	No access
0	0	1	0	(Invalid combination)
0	0	0	1	(Invalid combination)
0	0	1	1	(Invalid combination)
0	1	0	0	(Invalid combination)
0	1	1	0	(Invalid combination)
0	1	0	1	(Invalid combination)
0	1	1	1	(Invalid combination)
1	0	0	0	Read
1	0	1	0	Read and Insert (perhaps for test bed operator)

1	0	0	1	Read and Delete (e.g. administrator)
1	0	1	1	Read, Delete and Insert (e.g. administrator)
1	1	0	0	Read and Update (e.g. engineer)
1	1	1	0	All except Delete (e.g. engineer)
1	1	0	1	All except Insert
1	1	1	1	All access rights

To avoid “blind access”, all combinations of Update, Insert and Delete rights without granting the Read right are considered to be invalid (Configuration tools should not accept such combinations). In case of such combinations, the server will deny access and return an error message.

Note: Every desired right needs to be entered separately. There is not a hierarchy of rights in the sense of “Right A automatically implies Right B”.

Hint for implementation:

- To avoid complex and unnecessary attribute protection, access to some particular attributes (IDs, references) should be denied by client application programs rather than via attribute protection. Client Log-on to the Server

- When a session is opened, the client logs on by indicating the userID and password. The server checks if the information is valid (please see section 9.8.11 “User Authentication”). If the information is invalid, an error message will be returned.

Note: For client requests during the same open session, it is no longer necessary for the client to provide userID and password.

9.8.1 AOP_SETSECURITYLEVEL

Defines what security-level is used on some element.

These types can be used (and mixed), defined by the parameter <seclevel>:

1.) No security on element (<seclevel> = 0)

2.) Element-Security switched on (protects the whole element, bit 0 set)

3.) Instance-Security switched on (protects each instance, bit 1 set)

4.) Attribute-Security switched on (protects the attributes, bit 2 set)

Definition of setType:

```
SetSecurityLevelRet  AOP_SetSecurityLevel(SetSecurityLevelReq) = 69;
```

```
enum AOP_SetType {             /* how to modify bitmasked values */
 SET,                          /* Set the value to new bits (clears all
                                  bits before) */
 ADD,                          /* Adds the new bits to the existing
                                  ones (OR-function) */
 REMOVE                        /* Removes the bits from the existing
                                  value (~AND function) */
};

struct SetSecurityLevelReq {  /* setsecuritylevel-request structure */
  AOP_Id      envId;          /* environ. handle */
  AOP_Id      applId;         /* application element id */
  AOP_Flags   seclevel;       /* security level (bitmasked) */
  AOP_SetType setType;        /* set/add/remove security level bits */
};

union SetSecurityLevelRet switch (AOP_Status retState) {
                              /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Flags seclevel;/* return security-level */
};
```

9.8.2 AOP_GETSECURITYLEVEL

Get the current security-level of the specified element-id.

```
GetSecurityLevelRet  AOP_GetSecurityLevel(GetSecurityLevelReq) = 70;
struct GetSecurityLevelReq {  /* getsecuritylevel-request structure */
  AOP_Id        envId;        /* environ. handle */
  AOP_Id        applId;       /* application element id */
};

union GetSecurityLevelRet switch (AOP_Status retState) {
                              /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Flags seclevel;/* return security-level */
};
```

9.8.3 AOP_SETRIGHTS

Set access-rights (ACL) for some data-object(s) and usergroup(s)

```
SetRightsRet   AOP_SetRights(SetRightsReq) = 61;
```
```
struct SetRightsReq {     /* setrights-request structure */
  AOP_Id      envId;      /* environ. handle */
  AOP_Acs     acl<>;      /* ACL */
  AOP_SetType setType;    /* set/add/remove bits from rights in
                             ACL */
};

struct SetRightsRet {
  AOP_Status retState;
};
```

9.8.4 AOP_GETRIGHTS

Get ACL for some data-object(s) and usergroup(s)

```
GetRightsRet  AOP_GetRights(GetRightsReq) = 62;
```
```
struct GetRightsReq {     /* getrights-request structure */
  AOP_Id      envId;      /* environ. handle */
  AOP_ElemId  elemId<>;   /* Application element+instance Ids */
  AOP_Name    attrName<>; /* Application attribute names (list of
                             elements + list of attributes at the
                             same time is not allowed)*/
};

union GetRightsRet switch (AOP_Status retState) {
                          /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Acs acl<>; /* ACL will be returned */
};
```

9.8.5 AOP_SETINIRIGHTS

Define ACL-Templates (IRL) for some data-object(s) and usergroup(s).

```
SetIniRightsRet  AOP_SetIniRights(SetIniRightsReq) = 63;
```
```
struct SetIniRightsReq {  /* setinirights-request structure */
  AOP_Id       envId;     /* environ. handle */
  AOP_Irs      irl<>;     /* Initial rights list */
  AOP_SetType  setType;   /* set/add/remove bits from rights */
};

struct SetIniRightsRet {
  AOP_Status retState;
};
```

9.8.6 AOP_GETINIRIGHTS

Get ACL-Templates (IRL) for some data-object and usergroup(s)

```
GetIniRightsRet  AOP_GetIniRights(GetIniRightsReq) = 64;
struct GetIniRightsReq {  /* getinirights-request structure */
  AOP_Id      envId;      /* environ. handle */
  AOP_ElemId  elemId<>;   /* List of application element+instance
                             Ids */
};

union GetIniRightsRet switch (AOP_Status retState) {
                          /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Irs irl<>; /* Initial rights list will be returned */
};
```

9.8.7 AOP_SETINIRIGHTSREF

With this method an attribute (which must be a reference attribute to some application element) can be marked as a Initial-Rights-Reference-Attribute. At the time of inserting a new instance the value of this attribute will be resolved (reference to an element) and the IRL will be taken from the referenced instance. This is useful for storing data-depended IRLs, which can be referenced at the time of insertion.

```
SetIniRightsRefRet  AOP_SetIniRightsRef(SetIniRightsRefReq) = 65;
struct SetIniRightsRefReq {  /* setinirightsref-request structure */
  AOP_Id        envId;       /* environ. handle */
  AOP_Id        applId;      /* application element Id */
  AOP_Name      refAttr;     /* Reference Attribute */
  AOP_Flags     set;         /* use(1) or don't use(0) the
                                reference-attribute for ACL
                                templates */
};

union SetIniRightsRefRet switch (AOP_Status retState) {
                             /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Flags set;    /* set-value will be returned */
};
```

9.8.8 AOP_GETINIRIGHTSREF

This method returns a list of attributes that are marked as IRL-Reference-Attributes.

```
GetIniRightsRefRet  AOP_GetIniRightsRef(GetIniRightsRefReq) = 66;
---------------------------------------------------------------------
struct GetIniRightsRefReq {  /* getinirightsref-request structure */
  AOP_Id   envId;            /* environ. handle */
  AOP_Id   applId;           /* Application element Id */
};

union GetIniRightsRefRet switch (AOP_Status retState) {
                             /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Name attr<>;  /* attribute-list will be returned */
};
```

9.8.9 AOP_SETCURRENTINIRIGHTS

A client may define the ACL-Template by itself. This call defines an ACL that will be used in all subsequent insertions of the current session (parameter <set> must be 1). If parameter <set> is set to 0 (false), the prior defined ACL will not be used any longer in insertion-methods.

```
SetCurrentIniRightsRet
AOP_SetCurrentIniRights(SetCurrentIniRightsReq) = 67;
---------------------------------------------------------------------
struct SetCurrentIniRightsReq {  /* setcurrentinirights-request
                                    structure */
  AOP_Id    envId;               /* environ. handle */
  AOP_Irs   irl<>;               /* Initial ACL to use in subsequent
                                    creation of new instances  */
  AOP_Flags set;                 /* Use(1) this ACL or don't use(0)
                                    this ACL for further inserts
                                    anymore */
};

union SetCurrentIniRightsRet switch (AOP_Status retState) {
                                 /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Flags set;        /* set-value will be returned */
};
```

9.8.10 AOP_SETPASSWORD

Set a new password for some user.

If the currently logged in user is member of the superuser-group, he needs not to supply the current password in <oldpwd>. "Normal" users must supply the current password in <oldpwd> that they can change their own password to <newpwd>. "Normal" users cannot change passwords of other users, even if they supply the correct current password (of the other user) in <oldpwd>.

```
SetPasswordRet  AOP_SetPassword(SetPasswordReq) = 68;
```

```
struct SetPasswordReq {       /* setpassword-request structure */
  AOP_Id        envId;        /* environ. handle */
  AOP_Id        userid;       /* user-id to set the password for */
  AOP_String    oldpwd;       /* current password */
  AOP_String    newpwd;       /* new password */
};

union SetPasswordRet switch (AOP_Status retState) {
                              /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Flags ok;      /* ok=1 if password changed otherwise
                                 ok=0 */
};
```

9.8.11 USER AUTHENTICATION

Authentication of the username and password will be done by checking the Session-Parameter USER. The client has to perform an AOP_SetPar with the parameter USER and two values (list of strings, NULL-separated) <username> and <password> (see AOP_SetPar). The password has to supplied unencrypted. Encryption of the password will be done in the server (compatibility to ODS3.0).

9.8.12 PASSWORD ENCRYPTION IN ODS VERSION 5.0 (RPC)

GENERAL

Password encryption is a critical part of the user authentication, because passwords should never be stored in plain (unencrypted) form or with an encryption algorithm that allows decryption of the password. The algorithm must be standardized that means the whole algorithm must be made public, without the impact that someone could derive passwords from the encoded passwords.

MD5

The algorithm MD5, originally developed by the RSA Data Security, Inc. 1991, is a state of the art algorithm that belongs to the group of *hash functions or digital signatures*. MD5 can only encrypt data, but cannot decrypt it. The password can be checked by comparing the encrypted version of the user-supplied password to the stored (also encrypted) version in the database. If they match the passwords are identical.

HISTORY

MD2, MD4 and MD5 are message-digest algorithms developed by Rivest. They are meant for digital signature applications where a large message has to be "compressed" in a secure manner before being signed with the private key. All three algorithms take a message of arbitrary length and produce a 128-bit message digest. While the structures of these algorithms are somewhat similar, the design of MD2 is quite different from that of MD4 and MD5. MD2 was optimized for 8-bit machines, whereas MD4 and MD5 were aimed at 32-bit machines. **Description and source code for the three algorithms can be found as Internet RFCs 1319-1321**. MD2 was developed by Rivest in 1989. The message is first padded so its length in bytes is divisible by 16. A 16-byte checksum is then appended to the message, and the hash value is computed on the resulting message. Rogier and Chauvaud have found that collisions for MD2 can be constructed if the calculation of the checksum is omitted. This is the only cryptoanalytic result known for MD2.

MD4 was developed by Rivest in 1990. The message is padded to ensure that its length in bits plus 64 is divisible by 512. A 64-bit binary representation of the original length of the message is then concatenated to the message. The message is processed in 512-bit blocks in the Damgård/Merkle iterative structure, and each block is processed in three distinct rounds. Attacks on versions of MD4 with either the first or the last rounds missing were developed very quickly by Den Boer, Bosselaers and others. Dobbertin has shown how collisions for the full version of MD4 can be found in under a minute on a typical PC. In recent work, Dobbertin (Fast Software Encryption, 1998) has shown that a reduced version of MD4 in which the third round of the compression function is not executed but everything else remains the same, is not one-way. Clearly, MD4 should now be considered broken.

MD5 was developed by Rivest in 1991. It is basically MD4 with "safety-belts" and while it is slightly slower than MD4, it is more secure. The algorithm consists of four distinct rounds, which has a slightly different design from that of MD4. Message-digest size, as well as padding requirements, remain the same. Den Boer and Bosselaers have found pseudo-collisions for MD5. More recent work by Dobbertin has extended the techniques used so effectively in the analysis of MD4 to find collisions for the compression function of MD5. While stopping short of providing collisions for the hash function in its entirety this is clearly a significant step. Van Oorschot and Wiener have considered a brute-force search for collisions in hash functions, and they estimate a collision search machine designed specifically for MD5 (costing $10 million in 1994) could find a collision for MD5 in 24 days on average. The general techniques can be applied to other hash functions.

HASH FUNCTIONS

A *hash function H* is a transformation that takes an input m and returns a fixed-size string, which is called the hash value *h* (that is, *h* = *H*(*m*)). Hash functions with just this property have a variety of general computational uses, but when employed in cryptography, the hash functions are usually chosen to have some additional properties.

The basic requirements for a cryptographic hash function are as follows.

- The input can be of any length.
- The output has a fixed length.
- *H*(*x*) is relatively easy to compute for any given *x*.
- *H*(*x*) is one-way.
- *H*(*x*) is collision-free.

A hash function *H* is said to be *one-way* if it is hard to invert, where "hard to invert" means that given a hash value *h*, it is computationally infeasible to find some input *x* such that *H*(*x*) = *h*. If, given a message *x*, it is computationally infeasible to find a message *y* not equal to *x* such that *H*(*x*) = *H*(*y*), then *H* is said to be a *weakly collision-free* hash function. A *strongly collision-free* hash function *H* is one for which it is computationally infeasible to find any two messages *x* and *y* such that *H*(*x*) = *H*(*y*).

The hash value represents concisely the longer message or document from which it was computed; this value is called the *message digest*. One can think of a message digest as a "digital fingerprint" of the larger document.

Perhaps the main role of a cryptographic hash function is in the provision of message integrity checks and digital signatures. Since hash functions are generally faster than encryption or digital signature algorithms, it is typical to compute the digital signature or integrity check to some document by applying cryptographic processing to the document's hash value, which is small compared to the document itself. Additionally, a digest can be made public without revealing the contents of the document from which it is derived. This is important in digital timestamping where, using hash functions, one can get a document timestamped without revealing its contents to the timestamping service.

9.9 ERROR HANDLING

There are two kinds of errors: Failure on RPC (e.g. connection broken, using incompatible requests resulting in *Can't decode argument*s, timeouts, ...) and application level failures (e.g. missing mandatory attribute, application element not found, ...). Clients can detect an error when they get a NULL pointer returned instead of data. The RPC error reason can be determined by aop_geterr. A limited number of union discriminators are used which may be expanded in future.

```
enum AOP_Status {        /* error codes */
       S_OK      =  0,   /* OK */
                         /*   version 2 states */
       E_BSS     =  1,   /* basic system error state */
       W_BSS     =  2,   /* basic system warning state */
       E_SSS     =  3,   /* --"-- error */
       W_SSS     =  4,   /* subsystem (e.g. FORINT, RDBMS) warning */

                         /* version 1 states and version 2 states not
                            yet moved to new state interface; states
                            which havn't been used are removed */

       E_NOTOPEN = 13,   /* not yet open, environment */
       E_DAC     = 21,   /* data access error */
                         /* for detail see return structure */
                         /* Version 5.0 states */
       E_SECURITY= 41,   /* security violation */

       E_MISC    = 99    /* to be defined */

};
```

9.9.1 AOP_GetErr

Get error details

```
GetErrRet AOP_GetErr(GetErrReq) = 59;
```

```
struct GetErrReq {
  AOP_Id        envId;                 /* environ. handle */
};

union GetErrRet switch (AOP_Status retState) {
  case S_OK:       void;
  case E_NOTOPEN:  void;
  case E_SECURITY: void;
  case E_BSS: AOP_String      ebsMsg;
  case W_BSS: AOP_String      wbsMsg;
  case E_SSS: AOP_String      essMsg;
  case W_SSS: AOP_String      wssMsg;
  case E_MISC: AOP_String     errMsg;
  case E_DAC: AOP_DAC_Status  dacCode;
};
```

With a state `W_xxx`, details gives the warning text (see e.g. aop_getvalval), with `E_xxx` a textual error message is available (e.g. the Oracle error message). `E_DAC` includes a numeric subcode. See the request definitions which states may be returned.

9.10 RESTRICTIONS OF THE RPC-API

The types listed in the seq_rep_enum are not fully supported by the RPC-API. Only the types

- explicit
- implicit_constant
- implicit_linear
- implicit_saw

are supported. This is due to the fact that there are only two interface parameters available to select between implicit and explicit on one hand and between constant, linear, and saw on the other hand.

9.11 EXAMPLE CALLING SEQUENCE

EXAMPLE:

This is an example how a client could talk to a server. To keep it simple, the parameters have been omitted.

```
// Open Session
AOP_Openenv      //returns Environment-Id

// Get Metainformation (elements, attributes, references)
AOP_GetApplInf   //returns MetaInformation

// Do some work with attrtibutes in elements…
AOP_GetVal       //returns attribute-values (see also AOP_GetValE)
AOP_PutVal       //modify, insert or delete attribute-values

// Do some work with measurement values (partial matrixes)
AOP_GetValAttr   //returns partial matrix structure (attributes)
AOP_GetValVal    //returns partial matrix
AOP_PutValVal    //modify, insert or delete values in partial matrix

//Close Session
AOP_CloseEnv     //Closes the session
```

9.12 ASAM ODS VERSION 5.0 RPC-API IDL

```
/***************************************************************************
 *
 *  ASAM - ODS 5.0 client server ONC RPC interface definition
 *
 *
 *  2003/06/02 Revision 4.2: Added AoModelMapper, AoLog       Gerald Sammer
 *
 ***************************************************************************
 */

/*
 *  The interface-definition is based on ONC. The necessary tools for
 *  creation of C-headerfiles, client and server stubs and conversion
 *  functions (xdr) is free available (RPCGEN)
 *
 *  For any method available thru the protocol level (client server)
 *  the interface is defined by setting up request and return
 *  structures.
 *
 *  Version support
 *  ---------------
 *  This server supports version 3.0, 3.2 and ODS 5.0/RPC
 *
 *
 */
```

```
/* ============================================== platform adaptions
 * adapt aods.h for OS - dependencies to make aods.x more portable
 * Note that rpcgen uses the local cpp (preprocessor)
 */
%
%/* ------------------begin of section defined by aods.x ----------------
% */
#ifdef RPC_HDR
%#define _AODS_INTERFACE_VERSION "$Revision: 3.2 $"
%#ifndef _AODS_H
%#define _AODS_H
%#endif
%
#ifdef _WIN32
%/* remove definitions used by Visual C++ on NT
% * and add winsock.h if not using GNU on WIN32
% */
%#include <rpc/rpc.h>
#ifndef __CYGWIN32__
%#include <winsock.h>
%#undef S_OK
%#undef E_NOTIMPL
#endif
#endif

#ifdef hpux
%/* Do not edit, automatically created by rpcgen
% * the HP-UX rpcgen doesnt include rpc.h into aods.h but into
% * the .c files created -> force to put it into aods.h if not yet done
% * since VMS comes without rpcgen hpux is used for VMS too
% * but dont forget to remove the include <rpc/rpc.h> line from aods_xdr.c
% * and stubs manually (or by awk) when using those files on VMS
% */
%#ifdef vms
%#ifndef _RPCXDR_LOADED
%#include <ucx$rpcxdr.h>
%#endif
%#else
%#ifndef _STDLIB_INCLUDED
%#include <stdlib.h>
%#endif

#ifdef hpux10
%#ifndef _RPC_RPC_INCLUDED
#else
%#ifndef _RPC_TYPES_INCLUDED
#endif
%#include <rpc/rpc.h>
%#endif

%#endif
#endif
%
%extern unsigned long rpc_number;        /* RPC program number settable */
#endif

#ifdef RPC_SVC
%
%/* include authentication handling on server side
%*/
%#include "aods_auth.h"
#endif
```

```
#ifdef RPC_CLNT
%
%/* for some systems where rpcgen of foreign system is used struct timeval may
% * be undefined for TIMEOUT default definition
% */
%#include <time.h>
%
%/* nonstandard emulation of ListEnv
%*/
%#define AOP_ListEnv getenv ("AODSKNOWNENV")
%
#endif
%/* ---------------- end of aods.x defined section -----------------------
% */
%
```

```
/* =============================================== interface definition
 */

/* Some AODS data types
 * Any module may include the protocoll compiler resulting
 * header file and use those constants
 */

const MAXAOPNAMLEN = 128;
typedef string AOP_Name<MAXAOPNAMLEN>;
typedef AOP_Name AOP_NameSeq<>;

const MAXSTRINGLEN = 2000;       /* maximum string length */
typedef string AOP_String<MAXSTRINGLEN>;

const MAXBSTREAMLEN = 1048576;     /* maximum byte stream length */
typedef unsigned char bstream<MAXBSTREAMLEN>;

typedef long AOP_Id;            /* Identifiers/Handles always int */

typedef long AOP_Flags;         /* Used for Flags */

typedef short AOP_ValFlags<>;   /* Used for Flags to a Value */
```

```
/* =============================================== return states
 */
enum AOP_Status {       /* error codes */
        S_OK      =  0,         /* OK */
                                /*   version 2 states */
        E_BSS     =  1,         /* basic system error state */
        W_BSS     =  2,         /* basic system warning state */
        E_SSS     =  3,         /* --"-- error */
        W_SSS     =  4,         /* subsystem (e.g. FORINT, RDBMS) warning */

                                /* version 1 states and version 2 states
                                   not yet moved to new state interface
                                   states which havn't been used are removed */

        E_NOTOPEN = 13,         /* not yet open, environment */
        E_DAC     = 21,         /* data access error */
                                /* for detail see return structure */
        E_SECURITY= 41,         /* security violation */
    E_LICENSE = 42,           /* license error */
        E_MISC    = 99          /* to be defined */

};

enum AOP_DAC_Status {   /* subcodes used in version 1, replaced by x_SSS */
        DACINVATTR,             /* referenced attribute doesnt exist */
        DACNOPRIV,              /* privilege violation */
        DACMANDATTR,            /* mandatory attribute missing */
        DACCONSTRAINT,          /* constraint conflict */
        DACOTHER
};

/* =============================================== general type definitions
*/
enum AOP_DataType {     /* known data types */
        DT_RESERVED = 0,         /* 0 is called DT_UNKNOWN in the other chapters of
the ASAM ODS specification */
        DT_STRING = 1,           /*      string */
        DT_SHORT = 2,            /*      short integer */
        DT_FLOAT = 3,            /*      float */
        DT_BYTE = 5,             /*      byte */
        DT_LONG = 6,             /*      long integer */
        DT_DOUBLE = 7,           /*      double */
        DT_DATE = 10,            /*      date format */
        DT_BYTESTR = 11,         /*      byte string */
        /* blob holds beside the binary data (no architecture adaption)
         * a blob type which may be used to tell the client who may
         * interpret those data
         */
        DT_BLOB = 12             /* includes blob identifier */
};

/* These types are used to allow general purpose clients to interpret the
 * data
 */
enum AOP_BasElem {      /* Basic Entities */
        BE_RESERVED = 0,         /*  unknown base element */
        BE_ENV = 1,              /*      Environment */
        BE_TST = 2,              /*      Subtest */
        BE_MEA = 3,              /*      Measurement */
        BE_MEQ = 4,              /*      measured quantity */
        BE_DNA = 11,             /*      quantity name */
        BE_DNG = 12,             /*      quantity group */
        BE_DUN = 13,             /*      unit */
        BE_DUG = 14,            /*      unit group */
        BE_DIM = 15,            /*      physical dimension */
        BE_UUT = 21,            /*      UnitunderTest */
        BE_UUP = 22,            /*      --"-- part */
        BE_INS = 23,            /*      TestEquipment */
        BE_INP = 24,            /*      --"-- part */
        BE_SEQ = 25,            /*      Test sequence */
        BE_SEP = 26,            /*      --"-- part */
        BE_USR = 34,            /*      user */
        BE_UGR = 35,            /*      user group */
        BE_TEST = 36,           /*      Test */
        BE_TESTDEVICE = 37,     /*      Testdevice */
        BE_SUBMATRIX = 38,      /*      AoSubmatrix */
        BE_LOCALCOLUMN = 39,    /*      AoLocalColumn */
        BE_MODELMAPPER = 41,    /*      AoModelMapper */
        BE_LOG = 43,            /*      AoLog */
        BE_PARAMETER = 44,      /*      AoParameter */
        BE_PARAMATERSET = 45,   /*      AoParameterSet */
        BE_NAMEMAP = 46,        /*      AoNameMap */
        BE_ATTRIBUTEMAP = 47    /*      AoAttributeMap */
};


enum AOP_SelOpcode {    /* Type of selection */
        SOMATCH,                /*      match           ==      */
        SOUMATCH,               /*      unmatch         !=      */
        SORANGE,                /*      range           < x >   */
        SOLESS,                 /*      less            <       */
        SOGREATER,              /*      greater         >       */
        SOLESSEQ,               /*      less equal      <=      */
        SOGREATEREQ,            /*      greater equal   >=      */
        SOINSET,                /*      within set      in      */
        SONOTINSET,             /*      not within set  !in     */
        SOORDER,                /*      order by                */
        SOGROUP,                /*      group by                */
        SOINSENSITIVE,          /*      insensitive search      */
        SOLIKE,                 /*      like                    */
        SONOTLIKE,              /*      not like                */
        SONULL,                 /*      is NULL                 */
        SONOTNULL,              /*      is NOT NULL             */
        SOOPAND,                /*      operator AND            */
        SOOPOR,                 /*      operator OR             */
        SOOPNOT,                /*      operator NOT            */
        SOOPBOPEN,              /*      operator bracket open ( */
        SOOPBCLOSE              /*      operator bracket close )*/
};

/* see paper about multidimensional measured data arrays
 * for explanation on following flag
 */
enum AOP_IndepFl {      /* Flag for independent channel */
        INDNONE,                /*  no information (dependend) */
        INDEPEND,               /*  independent */
        SCALING                 /*  scales another column */
};

/* implicit means that the value may be calculated by a predefined
 * procedure. For implicit columns one (constant), two (start, increment) or
 * three values (start, increment, end) are stored within the column
 */
enum AOP_ImplFl {       /* Flag for implicit storage */
        IMPLNONE,               /*  explicit */
        IMPLICIT                /*  implicit */
};
```

```
enum AOP_SortDir {       /* Sort direction */
        ASCENDING,                /* lower to higher */
        DESCENDING                /* higher to lower */
};

enum AOP_JoinTyp {       /* way of joining partial mat. */
        JOIN,                     /* exact matches only */
        MERGE                     /* all */
};
```

```
/* ======================================================================
 *  General structures
 */

/* ------------------------- ElementId
 */
struct AOP_ElemId {     /* Unique identification of instance */
        AOP_Id  aid;             /* application element id */
        AOP_Id  iid;             /* instance id */
};
typedef AOP_ElemId AOP_ElemIdSeq<>;

/* -------------------------- ValMap
   This part describes the structures of an value map, this structure
   is common for all value accesses (Application Element Attribute Values
   and Local Column Values)
   Note that a string vector is a NULL separated byte array.
   The number of elements (strings) is equal to the
   number of NULL characters within the array. Therefor length is
   the number of bytes and not number of elements.
   BYTESTR is similar to STRING (each element may hold multible
   characters) except that instead of NULL-termination the count
   is included to allow binary data.
   Blop may be used only at places where no multible instanzes occur.

*/
struct Blob {
        char          bhdr<>;
        unsigned char bval<>;
};

union AOP_ValMap switch (AOP_DataType dtyp) {
  case DT_STRING:
        char    aval<>;          /* strings */
  case DT_SHORT:
        short   sval<>;          /* short integers */
  case DT_FLOAT:
        float   fval<>;          /* floats */
  case DT_LONG:
        long    lval<>;          /* long integers */
  case DT_DOUBLE:
        double  dval<>;          /* double floats */
  case DT_BYTE:
        unsigned char bval<>;    /* single bytes */
  case DT_DATE:
        char    tval<>;          /* date/time */
  case DT_BYTESTR:
        bstream cval<>;          /* counted byte streams */
  case DT_BLOB:
        Blob    blob;            /* never as array !!! */
  default:
        void;
};
/* -------------------------- Name + Unit
 */
struct AOP_NameU {
        AOP_Name    name;
        AOP_Id      unitId;
};
typedef AOP_NameU NUSeq<>;
/* -------------------------- Name + Unit + Value(s)
 */
struct AOP_NameV {
        AOP_Name     name;
        AOP_Id       unitId;
        AOP_ValMap   valMap;
};
typedef AOP_NameV NVSeq<>;
```

```
/* =====================================================================
 *  Request and return structure definition section
 */

/* -------------------------------- list environements
To emulate Master server (or service information brooker) an environement
variable on client side should be called (see above).
*/


/* -------------------------------- open environement
 * Note that current servers are restricted to one environement a time
 * Argument usage is free for implementors-
 * RDBMS implementation uses DBLOGON to (optionally) specify the
 * database logon string to be used.
 */

struct OpenEnvReq {       /* to open env. */
        AOP_NameV nvSeq<>;         /* environement open arguments */
};

union OpenEnvRet switch (AOP_Status retState) {
                          /* return value structure */
  case E_BSS: void;
  case E_SSS: void;
  case E_SECURITY: void;
  case E_MISC: void;
  case E_LICENSE: void;
  default: AOP_Id envId;            /* descriptor/handle/id whatever */
};

/* ------------------------------------ close environement
*/

struct CloseEnvReq  {     /* to close open env. */
        AOP_Id envId;
};

struct CloseEnvRet {
        AOP_Status retState;
};
```

```
/* --------------------------------- structure querry
   Whole basic structure has to be returned including all relations.
   This is a list of application elements (Identifier = AID) together with
   application name and type (basic element AOP_BasElem)
   In an additional list the relations (AOP_BasElem - pairs)
   will be returned
*/
struct GetApplInfReq {
  AOP_Id        envId;            /* environ. handle */
};

struct ApplInfSeq {       /* application element infos */
  AOP_Id        aiAId;            /* application element id */
  AOP_BasElem   aiBId;            /* basic element Id */
  AOP_Name      aiName;           /* application element name */
};

struct ApplRelSeq {       /* application relations infos */
  AOP_Id        arAId1;           /* from (0,1) resp. (n) */
  AOP_Id        arAId2;           /* to   (n)   resp. (m) */
  long          arRefNr;          /* attr number, 0 for n:m */
  AOP_Name      arName;           /* reference name */
  AOP_Flags     arConstr;         /* Constraint (e.g. not NULL) */
};

struct ApplInf {
  ApplInfSeq    aiSeq<>;         /* sequence of appelem info */
  ApplRelSeq    arSeq<>;         /* sequence of appref info */
};

union GetApplInfRet switch (AOP_Status retState) {
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: ApplInf  applInf;
};
```

```
/* -------------------------------- references (instance - level)
   This querry tries to get instances related to a given one thru
   allowed (see ApplInf) relations
*/

struct GetInstRefReq {
  AOP_Id        envId;           /* environ. handle */
  AOP_ElemId    elemId;          /* instance */
  AOP_Name      refName;
  AOP_Id        aId;             /* of type */
};



union GetInstRefRet switch (AOP_Status retState) {
                          /* return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_ElemId elemList<>;
};
```

```
/* ---- get attributes ------------------------------------------
   Structure to describe attributes of application elements.
   Applies to any entity also in many cases aBName matches aAName
   due to attrAName unchangeable on application level

   The request specifies the elementId.  The prototype uses only AId part
   of elementId (no instance attributes available)

*/

struct AttrSeq {           /* sequence of attributes */
  AOP_Name      aBName;           /* basic name */
                                  /* 0 if not basic */
  AOP_Name      aAName;           /* appl. name */
  AOP_DataType  aDataType;        /* attribute data type */
  AOP_Id        aUnit;            /* unit if global defined */
};


struct AttrInf {           /* attribute info including header */
  AOP_BasElem   aBId;             /* basic element Id */
  AOP_Id        aAId;             /* application element Id */
  AttrSeq       aSeq<>;
};

struct GetAttrReq {        /* request for attribute list */
  AOP_Id        envId;            /* environ. handle */
  AOP_ElemId    elemId;           /* application element */
};

union GetAttrRet switch (AOP_Status retState) {
                           /* return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AttrInf attrInf;        /* including entity */
};
```

```
/* ---------------------------- get values for application elements
   Querry structure, allows to specify ranges and patterns for querries.
   Values are always arrays. This allows i.e. on selection to pass 2 or
   more values for e.g.:        range from - to

   To cover the AODS requirement of making data access by explicitly
   specifying instance id (direct access to single element) it is
   necessary to define the selection elements AOP_NameS with
        name = <Idattr>
        selOpcode = SOMATCH
        value.AOP_ValMap_u.lval.lval[0] = <iid>
   IdAttr is the first aBName reportet using GetAttr

*/

struct AOP_NameS {        /* by value select */
  AOP_Name      name;            /* attribute name */
  AOP_ValMap    valMap;          /* range or pattern */
  AOP_SelOpcode selOpcode;       /* type of selection */
};
typedef AOP_NameS NSSeq<>;

struct GetValReq {        /* selection request structure */
  AOP_Id        envId;           /* environ. handle */
  AOP_Id        applId;          /* application element Id (from) */
  AOP_NameU     nuSeq<>;         /* sequence of to be reported attr's */
  AOP_NameS     nsSeq<>;         /* sequence of select fields */
  AOP_ElemId    elemId;          /* child/parent reference (where) */
  AOP_Name      refName;         /* named reference identifier */
};

union GetValRet switch (AOP_Status retState) {
                          /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_NameV nvSeq<>;    /* sequence of resulting attributes */
};
```

```
/* ----------------------------- enhanced get values for application elements
   Enhanced Query structure, allows
   - joins of application elements
   - to specify ranges and patterns (Values are always arrays.
                                     This allows i.e. on selection to pass 2 or
                                     more values for e.g.: in/notin, range from -
to)
   - to group selections with AND, OR and brackets (open, close)
   - to sort the result (order by)
   - to specify wildcards (including escape character)
   - partial access of the result (e.g. first 100, next 100, ..)

   To cover the AODS requirement of making data access by explicitly
   specifying instance id (direct access to single element) it is
   necessary to define the selection elements AOP_AIDNameS with
      selOpcode = SOMATCH
        AID = <application element ID>
        name = <Idattr>
        value.AOP_ValMap_u.lval.lval[0] = <iid>
   IdAttr is the first aBName reportet using GetAttr

*/
/* -------------------------- AID + Name + Unit
 */
struct AOP_AIDNameU {
  AOP_Id        aid;             /* application element ID */
  AOP_Name      name;            /* attribute name */
  AOP_Id        unitId;          /* requested unit */
};
typedef AOP_AIDNameU ANUSeq<>;

/* -------------------------- AID + Name + Unit + Value(s)
 */
struct AOP_AIDNameV {
  AOP_Id        aid;             /* application element ID */
  AOP_Name      name;            /* attribute name */
  AOP_Id        unitId;          /* requested unit */
  AOP_ValMap    valMap;          /* values */
};
typedef AOP_AIDNameV ANVSeq<>;

enum AOP_ReportListType { /* type of report list */
  RLCOMPLETE,                    /* all attributes */
  RLSELECTIVE                    /* given list of attributes */
};

union AOP_ReportList switch (AOP_ReportListType rlType) {
  case RLCOMPLETE:
    AOP_Id       aidSeq<>;  /* list of application elements - all attributes will
be reported */
  case RLSELECTIVE:
    AOP_AIDNameU anuSeq<>;  /* list of attribute to be reported */
  default: void;
};

struct AOP_AIDNameS {     /* by value select */
  AOP_SelOpcode selOpcode;       /* type of selection */
  AOP_Id        aid;             /* application element ID */
  AOP_Name      name;            /* attribute name */
  AOP_ValMap    valMap;          /* range or pattern */
  AOP_SortDir   dir;             /* sort order */
};
typedef AOP_AIDNameS ANSSeq<>;
```

```
struct AOP_RefDef {        /* reference definition */
  AOP_Id        fromAid;          /* start point (AID) of reference */
  AOP_ElemId    elemId;           /* child/parent reference (where) */
  AOP_Name      refName;          /* named reference identifier */
};

enum AOP_JoinType {        /* join type */
  JTDEFAULT,                      /* force inner join */
  JTOUTER                         /* force outer join on destination AID */
};

struct AOP_JoinDef {       /* join definition */
  AOP_Id        fromAid;          /* start point (AID) of reference */
  AOP_Id        toAid;            /* destination (AID) */
  AOP_Name      refName;          /* named reference identifier */
  AOP_JoinType  joinType;         /* join type */
};

enum AOP_ReqType {         /* type of request */
  RTDEFAULT,                      /* force query and close cursor */
  RTOPEN,                         /* force query and keep cursor open */
  RTCONTINUE,                     /* continue reading cursor */
  RTCLOSE                         /* close open cursor */
};

struct GetValEReq {        /* enhanced query request structure */
  AOP_Id         envId;            /* environ. handle */
  AOP_ReqType    reqType;          /* request type */
  AOP_ReportList repList;          /* definition of report list */
  AOP_AIDNameS   nsSeq<>;          /* sequence of conditions */
  AOP_JoinDef    jdSeq<>;          /* sequence of join definitions */
  AOP_RefDef     refDef;           /* reference definition */
  long           rowCnt;           /* number of requested rows
                                      - rowCnt<=0 report all rows */
};

union GetValERet switch (AOP_Status retState) {
                           /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: ANVSeq result<>;   /* sequence of resulting attribute sequences */
};
```

```
/* ----------------------------- get instance attribute information/values
   The result depends on the values of elemId. If only elemId.aid is set,
   AOP_GetInstAttr will return information (name, data type and unit but
   not the values) about all InstanceAttributes available for specified
   application element.
   If elemId referencing one instance (elemId.aid and elemId.iid are valid)
   all InstanceAttributes belonging to the specified instance will be
   returned.
*/

struct GetInstAttrReq {   /* selection request structure */
  AOP_Id        envId;           /* environ. handle */
  AOP_ElemId    elemId;          /* application element Id */
};

union GetInstAttrRet switch (AOP_Status retState) {
                          /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_NameV nvSeq<>;    /* sequence of resulting attributes */
};
```

```
/* ----------------------- insert/update/delete application element
 */
struct PutValReq {        /* update request structure */
  AOP_Id        envId;           /* environ. handle */
  AOP_ElemId    elemId;          /* application element Id */
  AOP_NameV     nvSeq<>;         /* sequence of values */
};

union PutValRet switch (AOP_Status retState) {
                          /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Id iid;            /* storage instance id */
};
```

```
/* ----------------------- for getting partial matrix info
*/

struct ValInfSeq {        /* partial matrix interrelations */
  AOP_Id        pmatId;
  AOP_Name      pmatName;        /* GIDAS requirement
                                    (sprintf of pmatId) */
  AOP_Name      presortNames <>; /* columns to sort in pmat */
  AOP_SortDir   presortDir;      /* sortdirection */
  AOP_Name      joinNames <>;    /* join columns */
  AOP_JoinTyp   joinTyp;
  AOP_Name      postsortNames <>; /* columns to sort on result */
  AOP_SortDir   postsortTyp;
};

struct GetValInfReq {
  AOP_Id        envId;
  AOP_ElemId    elemId;          /* aid has to be measurement */
};

union GetValInfRet switch (AOP_Status retState) {
                          /* info fetch return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: ValInfSeq iSeq<>;      /* sequence of info's */
};
```

```
/* ----------------------- get information on measured quantity/val
   The GetAttr/GetVal methods apply to the "general purpose" application
   elements  and not to the area of measurementpointnumbers, partial
   matrix. , measured quantity, ... (see AODS-datamodel).
   For this area AODS uses methods with applicationspecific behaviour.

   This method list measured quantity names and some attributes of them for
   one partial matrix.
   A list of partial matrices available may be fetch using GetValInf.
*/

struct ValAttrSeq {       /* linked list of attributes */
  AOP_Name      name;            /* appl. structure name */
  AOP_Id        qtyId;           /* reference to dict. (0 allowed) */
  AOP_Id        unitId;          /* UnitId */
  AOP_IndepFl   indepFl;         /* independent flag */
  AOP_ImplFl    implFl;          /* implicit flag */
  AOP_DataType  dataType;        /* attribute data type */
};

struct ValAttr {          /* measurement attribute cursor list */
  AOP_Id        pmatId;          /* PMatId */
  long          numPnt;          /* number of measurement point
                                    in partial matrix */
                                 /* should be extended for min/max */
  ValAttrSeq    vaSeq<>;         /* attribute sequence  */
};

struct GetValAttrReq {    /* request for attribute list */
  AOP_Id        envId;           /* environ. handle */
  AOP_ElemId    elemId;          /* measurement id */
  AOP_Id        pmatId;          /* partial matrix id (0 = all)*/
};

union GetValAttrRet switch (AOP_Status retState) {
                          /* return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: ValAttr valAttr<>;     /* column info for 1 or more PMat's */
};
```

```
/* ------------------------- partial matrix data (values)
   The range of data is selectable by specifying the column(s) (vrName)
   and the local measurement point number(s) (mPntNum mPntCnt)
*/

struct ValValSeq {       /* linked list of reported vectors */
  AOP_Name      name;             /*  MQ Name */
  AOP_Id        qtyId;            /* dictionary Id (ref) */
  AOP_Id        unitId;           /* unit Id */
  AOP_IndepFl   indepFl;          /* independent flag */
  AOP_ImplFl    implFl;           /* implicit (versus explicit) */
  AOP_ValMap    valMap;           /* vector for values */
  AOP_ValFlags  valFlags;         /* vector for values flags */
};

/* the partial matrix now holds 2 numbers to be able to preserve
 * origin of data
 */
struct ValVal {            /* local column values */
  AOP_Id        pmatId;           /* pMatId (0 .. yet undef) */
  long          mPntBase;         /* local point number base */
  long          mPntCnt;          /* # local points  */
  ValValSeq     vvSeq<>;          /* measurement value list */
};

struct GetValValReq {      /* request for measurement results */
  AOP_Id        envId;            /* environ. handle */
  AOP_ElemId    elemId;           /* elementId, bid to elemId.aid has to
                                     be BE_MEA */
  AOP_Id        pmatId;           /* partial matrix number */
  long          mPntBase;         /* from local measurement point number */
  long          mPntCnt;          /* number of measurement points requested */
  AOP_NameU     nuSeq<>;          /* sequence of attributes to report */
};

union GetValValRet switch (AOP_Status retState) {
                           /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: ValVal valVal<>;       /* partial matrices */
};

/* update/write/delete partial matrix -------------------------------------
 * usage: See programmers guide
 */

struct PutValValReq {    /* request for measurement results */
  AOP_Id        envId;            /* environ. handle */
  AOP_ElemId    elemId;           /* elementId, bid to elemId.aid has to
                                     be BE_MEA */
  ValVal        valVal<>;         /* sequence of headers + columns */
                                  /* (only one PMat for now per call) */
};

union PutValValRet switch (AOP_Status retState) {
                         /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
```

```
    case E_DAC: void;
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_Id pmatId;          /* pmatid given by server */
  };
```

```
/* ===================================================================
 * Connect two instances (n:m)
 */

struct SetInstRefReq {
  AOP_Id        envId;                  /* environ. handle */
  AOP_ElemId    elemId1;                /* one element */
  AOP_ElemId    elemId2;                /* the other */
  AOP_Name      refName;
  long          onoff;                  /* insert or remove */
};

struct SetInstRefRet {
   AOP_Status retState;
};
```

```
/* ===================================================================
 * Parametrization of ASAM services
 * This method is introduced to have a general purpose method to
 * influence behaviour of server by client including integration
 * of proprietary functions
 */

struct SetParReq {
  AOP_Id        envId;                  /* environ. handle */
  AOP_NameV     nvSeq<>;                /* parameters */
};

struct SetParRet {
   AOP_Status retState;
};

struct GetParReq {
  AOP_Id        envId;                  /* environ. handle */
  AOP_NameU     nuSeq<>;                /* parameter name-unit tuples */
};

union GetParRet switch (AOP_Status retState) {
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_NameV      nvSeq<>;      /* parameters */
};

/* extended error information
 */
struct GetErrReq {
  AOP_Id        envId;               /* environ. handle */
};

union GetErrRet switch (AOP_Status retState) {
  case S_OK:       void;
  case E_NOTOPEN:  void;
  case E_SECURITY: void;
  case E_BSS: AOP_String      ebsMsg;
  case W_BSS: AOP_String      wbsMsg;
  case E_SSS: AOP_String      essMsg;
  case W_SSS: AOP_String      wssMsg;
  case E_MISC: AOP_String     errMsg;
  case E_DAC: AOP_DAC_Status  dacCode;
};
```

```
/* ===================================================================
 * ASAM ODS Security methods
 * The following methods are used for get/set security on data
 * (Modifying ACL information).
 */

typedef long AOP_Rights;         /* Used for Rights-Values (bitmasked) */

enum AOP_SetType {               /* how to modify bitmasked values */
        SET,                     /* Set the value to new bits (clears all bits
before) */
        ADD,                     /* Adds the new bits to the existing ones (OR-
function) */
    REMOVE               /* Removes the bits from the existing value (~AND function)
*/
};

struct AOP_Acs {                 /* Access control structure */
  AOP_Id        groupId;        /* Usergroup Id */
  AOP_ElemId     elemId;           /* Application element+instance Id */
  AOP_Name attrName;         /* Application attribute name */
  AOP_Rights    rights;         /* rights-value (bitmasked) 5-bits */
};
typedef AOP_Acs AOP_Acl<>;

struct AOP_Irs {                 /* Initial rights structure */
  AOP_Id        groupId;        /* Usergroup Id */
  AOP_ElemId     elemId;           /* Application element+instance Id */
  AOP_Rights    rights;         /* rights-value (bitmasked) */
  AOP_Id         refAid;           /* referencing application element. If set to 0,
its been used by all elements that refer to it */
};
typedef AOP_Irs AOP_IrsSeq<>;

struct GetRightsReq {            /* getrights-request structure */
  AOP_Id        envId;          /* environ. handle */
  AOP_ElemId     elemId<>;         /* Application element+instance Ids */
  AOP_Name attrName<>;      /* Application attribute names (list of elements + list
of attributes
                                                          at the same time is not
allowed)*/
};

union GetRightsRet switch (AOP_Status retState) {
                    /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Acs acl<>;    /* ACL will be returned */
};

struct SetRightsReq {            /* setrights-request structure */
  AOP_Id        envId;           /* environ. handle */
  AOP_Acs  acl<>;        /* ACL */
  AOP_SetType   setType;        /* set/add/remove bits from rights in ACL */
};

struct SetRightsRet {
  AOP_Status retState;
};
```

```
struct GetIniRightsReq {          /* getinirights-request structure */
  AOP_Id        envId;          /* environ. handle */
  AOP_ElemId     elemId<>;      /* List of application element+instance Ids */
};

union GetIniRightsRet switch (AOP_Status retState) {
                          /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Irs irl<>;    /* Initial rights list will be returned */
};

struct SetIniRightsReq {          /* setinirights-request structure */
  AOP_Id        envId;            /* environ. handle */
  AOP_Irs  irl<>;        /* Initial rights list */
  AOP_SetType   setType;        /* set/add/remove bits from rights */
};

struct SetIniRightsRet {
  AOP_Status retState;
};

struct GetIniRightsRefReq {       /* getinirightsref-request structure */
  AOP_Id        envId;          /* environ. handle */
  AOP_Id   applId;              /* Application element Id */
};

union GetIniRightsRefRet switch (AOP_Status retState) {
                          /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Name attr<>;    /* attribute-list will be returned */
};

struct SetIniRightsRefReq {       /* setinirightsref-request structure */
  AOP_Id        envId;            /* environ. handle */
  AOP_Id        applId;           /* application element Id */
  AOP_Name      refAttr;          /* Reference Attribute */
  AOP_Flags     set;             /* use(1) or don't use(0) the reference-attribute
for ACL templates */
};

union SetIniRightsRefRet switch (AOP_Status retState) {
                          /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Flags set;        /* set-value will be returned */
};

struct SetCurrentIniRightsReq {  /* setcurrentinirights-request structure */
  AOP_Id        envId;            /* environ. handle */
  AOP_Irs       irl<>;           /* Initial ACL to use in subsequent creation of
new instances  */
  AOP_Flags     set;            /* Use(1) this ACL or don't use(0) this ACL for
further inserts anymore */
};

union SetCurrentIniRightsRet switch (AOP_Status retState) {
                          /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Flags set;         /* set-value will be returned */
};

struct SetPasswordReq {           /* setpassword-request structure */
  AOP_Id        envId;            /* environ. handle */
  AOP_Id        userid;           /* user-id to set the password for */
  AOP_String    oldpwd;           /* current password */
  AOP_String    newpwd;           /* new password */
};

union SetPasswordRet switch (AOP_Status retState) {
                          /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Flags ok;          /* ok=1 if password changed otherwise ok=0 */
};

struct SetSecurityLevelReq {      /* setsecuritylevel-request structure */
  AOP_Id        envId;            /* environ. handle */
  AOP_Id        applId;           /* application element id */
  AOP_Flags     seclevel;         /* security level (bitmasked) */
  AOP_SetType   setType;        /* set/add/remove security level bits */
};

union SetSecurityLevelRet switch (AOP_Status retState) {
                          /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
  case E_SECURITY: void;
  case E_MISC: void;
  default: AOP_Flags seclevel;    /* return security-level */
};

struct GetSecurityLevelReq {      /* getsecuritylevel-request structure */
  AOP_Id        envId;            /* environ. handle */
  AOP_Id        applId;           /* application element id */
};

union GetSecurityLevelRet switch (AOP_Status retState) {
                          /* select return value structure */
  case E_NOTOPEN: void;
  case E_BSS: void;
  case E_SSS: void;
  case E_DAC: void;
```

```
    case E_SECURITY: void;
    case E_MISC: void;
    default: AOP_Flags seclevel;   /* return security-level */
  };
```

```
/* ============================================================================
 * procedures
 */

program AODSPROG {

    version AODSVERS3 {

/* Open Environement ------------------------------------------------------   */
        OpenEnvRet              AOP_OpenEnv(OpenEnvReq)          = 1;

/* Close Environement ------------------------------------------------------  */
        CloseEnvRet             AOP_CloseEnv(CloseEnvReq)        = 2;

/* Metainfo - querries ----------------------------------------------------   */
        GetApplInfRet           AOP_GetApplInf(GetApplInfReq)    = 14;
        GetAttrRet              AOP_GetAttr(GetAttrReq)          = 12;

/* Metainfo - inserts -----------------------------------------------------   */

/* Data querry ------------------------------------------------------------- */
        GetValRet               AOP_GetVal(GetValReq)            = 21;
        GetInstAttrRet          AOP_GetInstAttr(GetInstAttrReq)  = 22;
        GetInstRefRet           AOP_GetInstRef(GetInstRefReq)    = 25;
        GetValERet              AOP_GetValE(GetValEReq)          = 26;

/* Data update on application elements -----------------------------------   */
        PutValRet               AOP_PutVal(PutValReq)            = 27;
        SetInstRefRet           AOP_SetInstRef(SetInstRefReq)    = 28;

/* Operations on measured values level --------------------------------      */
        GetValAttrRet           AOP_GetValAttr(GetValAttrReq)    = 31; /*
                    candidate for removal, use GetValVal with zero size      */
        GetValValRet            AOP_GetValVal(GetValValReq)      = 32;
        GetValInfRet            AOP_GetValInf(GetValInfReq)      = 35;
        PutValValRet            AOP_PutValVal(PutValValReq)      = 38;


/* Utility functions ------------------------------------------------------   */
        SetParRet                AOP_SetPar(SetParReq)            = 51;
        GetParRet                AOP_GetPar(GetParReq)            = 52;
        GetErrRet        AOP_GetErr(GetErrReq)           = 59;

/* Security functions -----------------------------------------------------   */
SetRightsRet             AOP_SetRights(SetRightsReq)                        = 61;
GetRightsRet             AOP_GetRights(GetRightsReq)                        = 62;
SetIniRightsRet          AOP_SetIniRights(SetIniRightsReq)                  = 63;
GetIniRightsRet          AOP_GetIniRights(GetIniRightsReq)                  = 64;
SetIniRightsRefRet       AOP_SetIniRightsRef(SetIniRightsRefReq)            = 65;
GetIniRightsRefRet       AOP_GetIniRightsRef(GetIniRightsRefReq)            = 66;
SetCurrentIniRightsRet   AOP_SetCurrentIniRights(SetCurrentIniRightsReq)    = 67;
SetPasswordRet           AOP_SetPassword(SetPasswordReq)                    = 68;
SetSecurityLevelRet      AOP_SetSecurityLevel(SetSecurityLevelReq)          = 69;
GetSecurityLevelRet      AOP_GetSecurityLevel(GetSecurityLevelReq)          = 70;
    } = 3;

/* Program number for all procedures --------------------------------------   */
} = rpc_number;
```

9.13 Revision History

Date Editor	Changes
2003-10-06 R. Bartz	Some errors have been corrected
2003-11-21 R. Bartz	A section on data type usage has been added
2003-12-30 R. Bartz	The **Release** version has been created
2004-03 R. Bartz	Section 9.10 "Restrictions of the RPC-API" has been included 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 10 ASAM ODS OO-API

Version 5.0

Association for **S**tandardization of
Automation and **M**easuring Systems

Dated: 30.09.2004

Status of Document

Reference:	ASAM ODS Version 5.0 ASAM ODS OO-API
Date:	30.09.2004
Author:	Hans-Joachim Bothe, HighQSoft; Dr. Helmut Helpenstein, National Instruments; Gerald Sammer, AVL; Karst Schaap, HighQSoft; Dr. Bruno Thelen, Schenck Pegasus
Type:	Specification
Doc-ID:	ASAM_ODS_50_CH10_OO_API.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 (odsftr@asam.net) to make sure this issue will be addressed within the next review cycle.

Contents

10 ASAM ODS OO-API 10-5

10.1 INTRODUCTION 10-5
10.2 INTERFACES OF THE OO-API 10-7
10.2.1 AOFACTORY 10-8
10.2.2 AOSESSION 10-13
10.2.3 APPLELEMACCESS 10-39
10.2.4 APPLICATIONATTRIBUTE 10-63
10.2.5 APPLICATIONELEMENT 10-91
10.2.6 APPLICATIONRELATION 10-125
10.2.7 APPLICATIONSTRUCTURE 10-143
10.2.8 BASEATTRIBUTE 10-177
10.2.9 BASEELEMENT 10-182
10.2.10 BASERELATION 10-190
10.2.11 BASESTRUCTURE 10-198
10.2.12 BLOB 10-212
10.2.13 COLUMN 10-220
10.2.14 ELEMRESULTSETEXTSEQITERATOR 10-232
10.2.15 ENUMERATIONDEFINITION 10-237
10.2.16 INSTANCEELEMENT 10-245
10.2.17 INSTANCEELEMENTITERATOR 10-277
10.2.18 MEASUREMENT 10-282
10.2.19 NAMEITERATOR 10-286
10.2.20 NAMEVALUEITERATOR 10-291
10.2.21 NAMEVALUEUNITIDITERATOR 10-296
10.2.22 NAMEVALUEUNITITERATOR 10-301
10.2.23 NAMEVALUEUNITSEQUENCEITERATOR 10-306
10.2.24 QUERY 10-311
10.2.25 QUERYEVALUATOR 10-318
10.2.26 SMATLINK 10-322
10.2.27 SUBMATRIX 10-334
10.2.28 VALUEMATRIX 10-337
10.3 ASAM ODS TYPE DEFINITIONS 10-361
10.3.1 ASAM ODS DATA TYPES 10-361
10.3.2 SUPPORTED DATA TYPE CONVERSIONS 10-362
10.3.3 ASAM ODS CONSTANTS 10-363
10.3.4 ASAM ODS ATTRIBUTE TYPES 10-364
10.3.5 ASAM ODS BUILD UP FUNCTIONS 10-364
10.3.6 ASAM ODS QUERY STATUS 10-364
10.3.7 ASAM ODS RIGHTS SET 10-364

10.3.8 ASAM ODS SELECT OPERATION CODE 10-365
10.3.9 ASAM ODS SELECT OPERATOR 10-365
10.3.10 ASAM ODS SET TYPE 10-366
10.3.11 ASAM ODS SEVERITY FLAG 10-366
10.3.12 ASAM ODS RELATIONS 10-366
10.3.13 ASAM ODS PATTERNS 10-369
10.3.14 ASAM ODS EXCEPTIONS 10-369
10.3.15 ASAM ODS SPECIFIC TYPES 10-373
10.3.16 ASAM ODS UNIONS 10-376
10.3.17 ASAM ODS STRUCTURES 10-379
10.4 PROGRAMMING EXAMPLES 10-388
10.4.1 ACCESSING THE ASAM ODS FACTORY OBJECT VIA CORBA 10-388
10.5 OO-API DEFINITION FILE 10-390
10.5.1 ODS.IDL 10-390
10.6 USING ASAM HARMONIZED DATATYPES 10-602
10.6.1 MAPPING FILE A_TYPES.IDL 10-602
10.7 QUICKREFERENCE 10-608
10.7.1 INTERFACE AOFACTORY 10-608
10.7.2 INTERFACE AOSESSION 10-608
10.7.3 INTERFACE APPLELEMACCESS 10-609
10.7.4 INTERFACE APPLICATIONATTRIBUTE 10-609
10.7.5 INTERFACE APPLICATIONELEMENT 10-610
10.7.6 INTERFACE APPLICATIONRELATION 10-611
10.7.7 INTERFACE APPLICATIONSTRUCTURE 10-611
10.7.8 INTERFACE BASEATTRIBUTE 10-612
10.7.9 INTERFACE BASEELEMENT 10-612
10.7.10 INTERFACE BASERELATION 10-612
10.7.11 INTERFACE BASESTRUCTURE 10-613
10.7.12 INTERFACE BLOB 10-613
10.7.13 INTERFACE COLUMN 10-613
10.7.14 INTERFACE ELEMRESULTSETEXTSEQITERATOR 10-613
10.7.15 INTERFACE ENUMERATIONDEFINITION 10-614
10.7.16 INTERFACE INSTANCEELEMENT 10-614
10.7.17 INTERFACE INSTANCEELEMENTITERATOR 10-615
10.7.18 INTERFACE MEASUREMENT 10-615
10.7.19 INTERFACE NAMEITERATOR 10-615
10.7.20 INTERFACE NAMEVALUEITERATOR 10-615
10.7.21 INTERFACE NAMEVALUEUNITIDITERATOR 10-616
10.7.22 INTERFACE NAMEVALUEUNITITERATOR 10-616
10.7.23 INTERFACE NAMEVALUEUNITSEQUENCEITERATOR 10-616
10.7.24 INTERFACE QUERY 10-616
10.7.25 INTERFACE QUERY 10-616
10.7.26 INTERFACE SMATLINK 10-617
10.7.27 INTERFACE SUBMATRIX 10-617
10.7.28 INTERFACE VALUEMATRIX 10-617
10.8 REVISION HISTORY 10-619

Scope

This document describes the OO-API of the ASAM ODS Version 5.0 with examples, necessary tables etc.

Intended Audience

This document is intended for implementers of ASAM ODS Version 5.0. It shall be used as a technical reference with examples how to use the required API methods for accessing all data stored compliant with 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

10 ASAM ODS OO-API

10.1 INTRODUCTION

The ASAM ODS OO-API (Object-Oriented Application Programmers Interface) is designed to allow easy and fine granular read/write access to any ASAM ODS data from a variety of programming languages and interpreters. Object oriented methods are the foundation of the ASAM ODS OO-API. Nevertheless non-object oriented programming languages can be used for application programming.

Current implementations of the OO-API base on CORBA as transport mechanism for objects and mediator for object access. However this is not prescribed by ASAM ODS. Any other approach for distributed object access may be implemented. This implicitly means that following this specification of the OO-API does not guarantee product compatibility. Therefore each server or client implementation must state what technology it is based on (e.g. CORBA version 2.0) and what third party software is required for a smooth operation (as it must e.g. state on what platforms it will run etc.). In case of doubt participating at any of the officially organized cross-tests may help to unveil possible compatibility issues. More information on cross-tests may be received from the ASAM e.V..

In contrary to the ASAM ODS RPC-API (see *chapter 9, RPC-API*), the fine granularity of the OO-API may potentially cause excessive network traffic even on small amounts of data. The main design goal of the ASAM ODS OO-API was programming convenience and ease of use; performance and network traffic optimizations have not been an important issue. However, implementations of the ASAM ODS OO-API may use the ASAM ODS RPC-API internally for better network performance.

The functionality of the ASAM ODS OO-API is precisely defined according the CORBA-IDL (Common Object Request Broker Architecture - Interface Definition Language). The CORBA-IDL was chosen for two reasons:
- Because it is a well-defined description language
- Because there are programs available that can generate CORBA stub- and skeleton-code for various languages (e.g. C++ and Java). OO-API language bindings are possible for languages like C, C++, Tcl, Perl, Python, Interactive Data Language (IDL), Visual Basic, Java, Smalltalk, Delphi and others.

The main advantages of this approach are easy implementation and simple function calls. No complex structures are used and only a few arguments are necessary per call. The arguments themselves have a simple structure (strings, longs, floats, string lists, value lists, etc.).

For details see the ASAM ODS definition file ODS.IDL (section 10.5).

The entry point of the OO-API object hierarchy is always the ASAM ODS factory object (AoFactory, see next section). As soon as an ASAM ODS factory object exists, it can be used to create an arbitrary number of ASAM ODS sessions (AoSession). Additionally, the factory object provides general information like the factory name and type, a description string and the current interface version.

Both AoFactory and AoSession, together with all required data type definitions and the definitions of the interfaces listed below are also part of the ODS.IDL:

- ApplicationStructure, ApplicationElement, ApplicationAttribute, ApplicationRelation
- ApplElemAccess
- BaseStructure, BaseElement, BaseAttribute, BaseRelation
- Blob
- Column
- ElemResultSetExtSeqIterator
- EnumerationDefinition
- InstanceElement, InstanceElementIterator
- Measurement
- NameIterator, NameValueIterator, NameValueUnitIdIterator, NameValueUnitIterator, NameValueUnitSequenceIterator
- Query, QueryEvaluator
- SMatLink
- SubMatrix
- ValueMatrix

The “Application*” interface group is used for creating, manipulating and querying information about the application model. Very often, it is not possible or desirable to change the application model. The application model is normally created in the design phase of a project and rarely modified thereafter, because modifications of the application model may result in major mapping efforts or even restructuring of the underlying database. The application model is a description of how the real data is organized and structured – thus the information in the application model is often referred to as meta-data.

The “ApplElemAccess” interface is used to access large amount of data within few client/server round-trips. By using this interface the performance can be improved for insert/delete/update operations. The “Base*” interface group is used for getting information about the ASAM ODS base model. The base model is read-only; there is no way for an application program to modify the information contained in this model.

The “Instance*” group is used to create, modify and delete data objects whose structure is defined in the application model.

Measurement, SubMatrix, SMatLink, ValueMatrix, Column and Blob objects are used for array data manipulation.

The Query Interface defines selections based on the application model. This object is essential for creating subsets of data structures, thus reducing the amount of required data. The design of the query interface intentionally is a flavor of the COS (OMG's **C**ommon **O**bject **S**ervices) Query Service. However, the Query interface is introduced but not yet supported in ODS 5.0 because the definition of the ASAM query language is not completed yet. Clients cannot use the interfaces Query, QueryEvaluator, and AoSession.createQueryEvaluator. An ASAM ODS Server Version 5.0 must throw an exception AO_NOT_IMPLEMENTED for all methods used in these interfaces.

Clients may only modify data within an active transaction. It is not allowed to insert, delete, or update data without transaction.

10.2 INTERFACES OF THE OO-API

On the following pages all callable objects, elements and functions of the OO-API are listed in alphabetical order. Additionally the calling sequences, parameters (with type), arguments, return types and returned values are listed respectively. Many examples in Java complete this description of the OO-API.

NOTE: The argument “pattern” is always case sensitive. “pattern” may have wildcard characters as follows: “?” for one matching character, “*” for a sequence of matching characters.

10.2.1 AOFACTORY

AOFACTORY_GETDESCRIPTION

Purpose

Get the description of the ASAM ODS factory. If the description is not available an empty string is returned and no exception is thrown. The server loads the description from the base attribute "description" of the instance of AoEnvironment.

Parameters

None.

Java Calling Sequence

T_STRING description = aoFactory.getDescription();

Returns:

Return-Name:	description
Return-Type:	T_STRING

The description of the ASAM ODS factory.

Examples:

Language: Java

```
String description;
description = aoFactory.getDescription();
...
```

Possible exceptions (for details see section 10.3.14)

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AOFACTORY_GETINTERFACEVERSION

Purpose

Get the interface version of the ASAM ODS factory. The interface version is for each ODS version a fixed string. The string for this version is 'OO-5.0'.

Parameters

None.

Java Calling Sequence

T_STRING interfaceVersion = aoFactory.getInterfaceVersion();

Returns:

Return-Name:	interfaceVersion
Return-Type:	T_STRING

The interface version of the ASAM ODS factory.

Examples:

Language: Java

```
String interfaceVersion;
interfaceVersion = aoFactory.getInterfaceVersion();
```

Possible exceptions (for details see section 10.3.14)

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AOFACTORY_GETNAME

Purpose

Get the name of the ASAM ODS factory. If the name is not available an empty string is returned and no exception is thrown.

Parameters

None.

Java Calling Sequence

T_STRING factoryName = aoFactory.getName();

Returns:

Return-Name:	factoryName
Return-Type:	T_STRING

The name of the ASAM ODS factory.

Examples:

Language: Java

```
String name;
name = aoFactory.getName();
```

Possible exceptions (for details see section 10.3.14)

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AOFACTORY_GETTYPE

Purpose

Get the type of the ASAM ODS factory. If the type is not available an empty string is returned and no exception is thrown. The server loads the type from the base attribute "Application_model_type" of the instance of AoEnvironment.

Parameters

None.

Java Calling Sequence

T_STRING factoryType = aoFactory.getType();

Returns:

Return-Name:	factoryType
Return-Type:	T_STRING

The type of the ASAM ODS factory.

Examples:

Language: Java

```
String type;
type = aoFactory.getType();
```

Possible exceptions (for details see section 10.3.14)

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AOFACTORY_NEWSESSION

Purpose

Establish a new session to an ASAM ODS server. The server normally checks the activity of the session and will close the session after a time period of inactivity.

Parameters

auth (Type = T_STRING)

A string that may contain authentication information. The following values are currently supported:

USER

PASSWORD

OPENMODE

The values may be specified in any order and have to be separated by comma.

Example:

"USER=hans, PASSWORD=secret, OPENMODE=read"

Java Calling Sequence

AoSession aoSession = aoFactory.newSession(auth);

Returns:

Return-Name:	aoSession
Return-Type:	AoSession

The new created ASAM ODS session.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
AoSession session;
session = aoFactory.newSession("USER=hans, PASSWORD=secret");
```

Possible exceptions (for details see section 10.3.14)

AO_CONNECT_FAILED

AO_CONNECT_REFUSED

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_OPEN_MODE_NOT_SUPPORTED

AO_SESSION_LIMIT_REACHED

10.2.2 AoSession

AoSession_abortTransaction

Purpose

Abort (rollback) a transaction. The changes made in the transaction are lost.

Parameters

None.

Java Calling Sequence

aoSession.abortTransaction();

Returns:

None.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");
  // Session successfully created ?
  if (session != null) {
     // Start a new transaction.
     session.startTransaction();

          ...

     // Abort transaction.
     session.abortTransaction();
     // Close the session.
     session.close();
  }
```

Possible exceptions (for details see section 10.3.14)

AO_ACCESS_DENIED

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

AOSESSION_CLOSE

Purpose

Close session to an ASAM ODS server. Active transactions are committed.

Parameters

None.

Java Calling Sequence

aoSession.close();

Returns:

None.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");
  // Session successfully created ?
  if (session != null) {
     ...
     // Close the session.
     session.close();
  }
```

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AoSession_commitTransaction

Purpose

Commit a transaction. The changes made in the transaction become permanent.

Parameters

None.

Java Calling Sequence

aoSession.commitTransaction();

Returns:

None.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");
  // Session successfully created ?
  if (session != null) {
     // Start a new transaction.
     session.startTransaction();

            ...

     // Commit changes.
     session.commitTransaction();
     // Close the session.
     session.close();
  }
```

Possible exceptions (for details see section 10.3.14)

AO_ACCESS_DENIED

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

AOSESSION_CREATEBLOB

Purpose

Create a new object with the Interface Blob on the server. This object can be used to create an attribute value of the datatype DT_BLOB.

Parameters

None.

Java Calling Sequence

Blob blob = aoSession.createBlob();

Returns:

Return-Name:	blob
Return-Type:	Blob

The reference of the blob object which is generated at the server

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AoSession_createQueryEvaluator

Purpose

Create a QueryEvaluator object.

Parameters

None.

Java Calling Sequence

```
QueryEvaluator queryEvaluator = aoSession.createQueryEvaluator();
```

Returns:

Return-Name:	queryEvaluator
Return-Type:	QueryEvaluator

The new created query evaluator object

Possible exceptions (for details see section 10.3.14)

AO_ACCESS_DENIED

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AOSESSION_FLUSH

Purpose

Make the changes permanent.

Parameters

None.

Java Calling Sequence

aoSession.flush();

Returns:

None.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");
  // Session successfully created ?
  if (session != null) {
     // Start a new transaction.
     session.startTransaction();
           ...
     // Flush changes
     session.flush();
           ...
     // Commit changes.
     session.commitTransaction();
     // Close session.
     session.close();
  }
```

Possible exceptions (for details see section 10.3.14)

AO_ACCESS_DENIED

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

AoSession_getApplElemAccess

Purpose

Get the application element access object from the current session.

Parameters

None.

Java Calling Sequence

ApplElemAccess applElemAccess = aoSession.getApplElemAccess();

Returns:

Return-Name:	applElemAccess
Return-Type:	ApplElemAccess

The application element access object.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.ApplElemAccess;
  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");
  // Session successfully created ?
  if (session != null) {
     // Get ApplElemAccess.
          ApplElemAccess aea = session.getApplElemAccess();

     ...

     // Close the session.
     session.close();
  }
```

Possible exceptions (for details see section 10.3.14)

AO_ACCESS_DENIED

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AOSESSION_GETAPPLICATIONSTRUCTURE

Purpose

Get the application model from the current session by returning an object with the interface ApplicationStructure. The complete information on the application model is available through that interface.

Parameters

None.

Java Calling Sequence

ApplicationStructure applicationStructure = aoSession.getApplicationStructure();

Returns:

Return-Name:	applicationStructure
Return-Type:	ApplicationStructure

The application model.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.ApplicationStructure;
  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");
  // Session successfully created ?
  if (session != null) {
     // Get Application Model.
          ApplicationStructure as =
           session.getApplicationStructure();

     ...

     // Close the session.
     session.close();
  }
```

Possible exceptions (for details see section 10.3.14)

AO_ACCESS_DENIED

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AoSession_getApplicationStructureValue

Purpose

Get the application model as values from the current session.

Parameters

None.

Java Calling Sequence

```
ApplicationStructureValue applicationStructureValue =
aoSession.getApplicationStructureValue();
```

Returns:

Return-Name:	applicationStructureValue
Return-Type:	ApplicationStructureValue

The application model as value.

Possible exceptions (for details see section 10.3.14)

AO_ACCESS_DENIED

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AOSESSION_GETBASESTRUCTURE

Purpose

Get the ASAM ODS base model from the current session by returning an object with the interface BaseStructure. The complete information on the base model is available through that interface; it includes all possible base elements with all possible base attributes as specified by ASAM ODS.

Parameters

None.

Java Calling Sequence

```
BaseStructure baseStructure = aoSession.getBaseStructure();
```

Returns:

Return-Name:	baseStructure
Return-Type:	BaseStructure

The base model.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AOSESSION_GETCONTEXT

Purpose

Get context variables from the session. A pattern string can be specified to select groups of variables.

Parameters

varPattern (Type = Pattern)

The name or the search pattern for the context variable(s).

Java Calling Sequence

NameValueIterator nvIterator = aoSession.getContext(varPattern);

Returns:

Return-Name:	nvIterator
Return-Type:	NameValueIterator

A list of context variables.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.NameValue;
  import org.asam.ods.NameValueIterator;
  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");
  // Session successfully created ?
  if (session != null) {
     // Create holder for return values.
     NameValueIterator      nvIte;
     NameValue[]            nvSeq;
     // Ask for all variables with any name pattern.
     nvIte = session.getContext("*");
     // Get max. 40 variables
     nvSeq = nvIte.nextN(40);
     ...
     // Print the result variable list to standard output.
     for (int i=0; i<nvSeq.length; i++) {
        System.out.println (nvSeq[i].name + " = \"" +
        nvSeq[i].value.u.stringVal() + "\"");
     }
     ...
     // Close the session.
     session.close();
  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_FOUND

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AoSession_getContextByName

Purpose

Get a context variable by its name from the session.

Parameters

varName (Type = Name)

The name of the requested context variable.

Java Calling Sequence

NameValue contextVariable = aoSession.getContextByName(varName);

Returns:

Return-Name:	contextVariable
Return-Type:	NameValue

The requested context variable.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");
  // Session successfully created ?
  if (session != null) {
     // Get a test variable named TestVar.
     NameValue nvPair = session.getContextByName("TestVar");
     ...
     // Close the session.
     session.close();
  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_FOUND

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AOSESSION_GETDESCRIPTION

Purpose

Get the description of the ASAM ODS session.The description of the session is identical with description of the ASAM ODS factory. If the description is not available an empty string is returned and no exception is thrown. The server loads the description from the base attribute "description" of the instance of AoEnvironment.

Parameters

None.

Java Calling Sequence

T_STRING description = aoSession.getDescription();

Returns:

Return-Name:	description
Return-Type:	T_STRING

The description of the ASAM ODS session.

Examples:

Language: Java

```
String description;
description = aoSession.getDescription();
...
```

Possible exceptions (for details see section 10.3.14)

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_CONNECTION_LOST

AOSESSION_GETLOCKMODE

Purpose

Get the current lock mode. The lock mode tells the server which objects to lock for upcoming changes. Application elements, instance elements or children of elements can be locked.

Parameters

None.

Java Calling Sequence

LockMode lockMode = aoSession.getLockMode();

Returns:

Return-Name:	lockMode
Return-Type:	T_SHORT

The current lock mode. The lock mode constants are defined in the interface LockMode. The interface definition language IDL does not allow to set the values of enumerations thus the constant definitions had to be done in an interface.

Possible exceptions (for details see section 10.3.14)

AO_ACCESS_DENIED

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

AOSESSION_GETNAME

Purpose

Get the name of the ASAM ODS session. The name of the session is identical with the name of the ASAM ODS factory. If the name is not available an empty string is returned and no exception is thrown. The server loads the description from the base attribute "name" of the instance of AoEnvironment.

Parameters

None.

Java Calling Sequence

Name sessionName = aoSession.getName();

Returns:

Return-Name:	sessionName
Return-Type:	Name

The name of the ASAM ODS session.

Examples:

Language: Java

```
String name;
name = aoSession.getName();
```

Possible exceptions (for details see section 10.3.14)

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_CONNECTION_LOST

AoSession_getType

Purpose

Get the type of the ASAM ODS session. The type of the session is identical with the type of the ASAM ODS factory. If the type is not available an empty string is returned and no exception is thrown. The server loads the type from the base attribute "Application_model_type" of the instance of AoEnvironment.

Parameters

None.

Java Calling Sequence

T_STRING sessionType = aoSession.getType();

Returns:

Return-Name:	sessionType
Return-Type:	T_STRING

The type of the ASAM ODS session.

Examples:

Language: Java

```
String type;
type = aoSession.getType();
```

Possible exceptions (for details see section 10.3.14)

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_CONNECTION_LOST

AOSESSION_LISTCONTEXT

Purpose

List the names of context variables from the session. A pattern string can be specified to select groups of variables.

Parameters

varPattern (Type = Pattern)

The name or the search pattern for the context variable(s).

Java Calling Sequence

```
NameIterator nameIterator = aoSession.listContext(varPattern);
```

Returns:

Return-Name:	nameIterator
Return-Type:	NameIterator

A list of context variable names.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NOT_FOUND

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AoSession_removeContext

Purpose

Remove context variables from the session. A pattern string can be specified to remove groups of variables.

Parameters

varPattern (Type = Pattern)

The name or the search pattern for the context variable(s) to be removed.

Java Calling Sequence

aoSession.removeContext(varPattern);

Returns:

None.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");
  // Session successfully created ?
  if (session != null) {
     // Create a test variable named TestVar.
     String varName  = "TestVar";
     String varValue = "Value of TestVar";
          session.setContextString(varName,varValue);
     ...
     // Remove context variable TestVar.
          session.removeContext(varName);
     ...
     // Close the session.
     session.close();
  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NOT_FOUND

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AOSESSION_SETCONTEXT

Purpose

Set/modify a known context variable or add a new context variable to the session.

Parameters

contextVariable (Type = NameValue)

The context variable.

Java Calling Sequence

aoSession.setContext(contextVariable);

Returns:

None.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.types.TS_Union;
  import org.asam.ods.types.TS_Value;
  import org.asam.ods.types.NameValue;

  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");
  // Session successfully created ?
  if (session != null) {

     // Define variable name and value.
     String   varName  = "TestVar";
     float    varValue = 4711.3;

     /* Build the name-value pair.
     short[]   flag   = {0};
          TS_Union  union  = new TS_Union();
     union.floatVal(varValue);
     TS_Value  tsVal  = new TS_Value(union,flag);
     NameValue nvPair = new NameValue(varName,tsVal)
     // Create a float test variable named TestVar.
          session.SetContext(nvPair);

     ...

     // Close the session.
     session.close();

  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AOSESSION_SETCONTEXTSTRING

Purpose

Set/modify a known context variable or add a new context variable to the session. This is a convienience method for the frequently used string variable type. It uses setContext internally.

Parameters

varName (Type = Name)

The name of the context variable.

value (Type = T_STRING)

The value of the context variable.

Java Calling Sequence

aoSession.setContextString(varName,value);

Returns:

None.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");
  // Session successfully created ?
  if (session != null) {
     // Create a test variable named TestVar.
     String varName  = "TestVar";
     String varValue = "Value of TestVar";
           session.setContextString(varName,varValue);
     ...
     // Close the session.
     session.close();
  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AOSESSION_SETCURRENTINITIALRIGHTS

Purpose

Every new created instance will set its initial rights to <acl> . This method overrides the default-methods for applying initial rights. The initial rights are only valid for the current session.

Parameters

irlEntries (Type = InitialRightSequence)

The current initial rights.

set (Type = T_BOOLEAN)

Set (1) or remove (0) the current initial rights. The previous initial rights get lost. If the parameter set is 0 (remove) the parameter irlEntries will be ignored.

Java Calling Sequence

```
aoSession.setCurrentInitialRights(irlEntries,set);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_ACCESS_DENIED

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

AOSESSION_SETLOCKMODE

Purpose

Set the new lock mode. The lock mode tells the server which objects to lock for upcoming changes. Application elements, instance elements or children of elements can be locked.

Parameters

lockMode (Type = T_SHORT)

The new lock mode. The lock mode constants are defined in the interface LockMode. The interface definition language IDL does not allow to set the values of enumerations thus the constant definitions had to be done in an interface.

Java Calling Sequence

```
aoSession.setLockMode(lockMode);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_ACCESS_DENIED

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

AOSESSION_SETPASSWORD

Purpose

Change the password for user defined by <username> to <newPassword>. A normal user must supply his current password <oldPassword>. The superuser can change the password without supplying the current password <oldPassword>. If no username is given the password of the user of te current session will be changed. The password is normally encrypted in the attribute of the user instance element. Creating a new user can be done by creating a new instance, afterwards the password must be set by the superuser.

Parameters

username (Type = T_STRING)

The name of the user for which the password will be changed. If no username is given the password of the current user will be changed. If the username differs from the current user the current user must be a superuser.

oldPassword (Type = T_STRING)

The current password of the user. A normal user must supply his current password. The superuser can change the password without supplying the current password.

newPassword (Type = T_STRING)

The new password of the user.

Java Calling Sequence

```
aoSession.setPassword(username,oldPassword,newPassword);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_ACCESS_DENIED

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

AO_WRONG_PASSWORD

AOSESSION_STARTTRANSACTION

Purpose

Start a transaction on the physical storage system (e.g. database system). Only when a transaction is started it is allowed to create or modify instances or measurement data. The changes get permanent with a commit of the transaction or will be lost with an abort of the transaction. If the session is closed the transaction will be committed automatically. If a transaction is already active an exception is thrown.

Parameters

None.

Java Calling Sequence

aoSession.startTransaction();

Returns:

None.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");
  // Session successfully created ?
  if (session != null) {
     // Start a new transaction.
     session.startTransaction();
           ...
     // Commit changes.
     session.commitTransaction();
     // Close session.
     session.close();
  }
```

Possible exceptions (for details see section 10.3.14)

AO_ACCESS_DENIED

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_ALREADY_ACTIVE

10.2.3 APPLELEMACCESS

APPLELEMACCESS_DELETEINSTANCES

Purpose

Delete instances from an application element. In case of inherited application elements the Id of the supertype has to be specified. This method can be used to delete several instances of the same application element, the method removeInstances removes one instance of an application element with the children of the instance element.

Parameters

aid (Type = T_LONGLONG)

The application element Id.

instIds (Type = S_LONGLONG)

The sequence of instance Id's. At the RPC-API this information was stored in the fields elemId and nvSeq of the structure PutValReq and the request AOP_PutValReq.

Java Calling Sequence

applElemAccess.deleteInstances(aid,instIds);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLELEMACCESS_GETATTRIBUTERIGHTS

Purpose

Retrieve access control list information for the given application attribute <aid>/<attr_name>.

Parameters

aid (Type = T_LONGLONG)

The Id of the application element.

attrName (Type = T_STRING)

The name of the attribute.

Java Calling Sequence

```
ACL[] aclEntries = applElemAccess.getAttributeRights(aid,attrName);
```

Returns:

Return-Name:	aclEntries
Return-Type:	ACLSequence

The access control list entries of the give application attribute.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLELEMACCESS_GETELEMENTINITIALRIGHTS

Purpose

Retrieve access control list information of the initial rights for the requested application element <aid>.

Parameters

aid (Type = T_LONGLONG)

The Id of the application element.

Java Calling Sequence

```
InitialRight[] initialRights = applElemAccess.getElementInitialRights(aid);
```

Returns:

Return-Name:	initialRights
Return-Type:	InitialRightSequence

The access control list entries of the given application element for the initial rights.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLELEMACCESS_GETELEMENTRIGHTS

Purpose

Retrieve access control list information for the requested application element <aid>.

Parameters

aid (Type = T_LONGLONG)

The Id of the application element.

Java Calling Sequence

```
ACL[] aclEntries = applElemAccess.getElementRights(aid);
```

Returns:

Return-Name:	aclEntries
Return-Type:	ACLSequence

The access control list entries of the given application element.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

ApplElemAccess_getInitialRightReference

<u>Purpose</u>

Get all attribute names (references) which are used to retrieve the Initial Rights.

<u>Parameters</u>

aid (Type = T_LONGLONG)

The application element Id.

<u>Java Calling Sequence</u>

```
Name[] refNameList = applElemAccess.getInitialRightReference(aid);
```

<u>Returns:</u>

Return-Name:	refNameList
Return-Type:	NameSequence

The names of the references which will be used for the initial rights determination.

<u>Possible exceptions (for details see section 10.3.14)</u>

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLELEMACCESS_GETINSTANCEINITIALRIGHTS

Purpose

Retrieve access control list information of the initial rights for the requested instance <aid>/<iid>.

Parameters

aid (Type = T_LONGLONG)

The Id of the application element.

iid (Type = T_LONGLONG)

The Id of the instance.

Java Calling Sequence

```
InitialRight[] initialRights = applElemAccess.getInstanceInitialRights(aid,iid);
```

Returns:

Return-Name:	initialRights
Return-Type:	InitialRightSequence

The access control list entries of the given instance for the initial rights.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLELEMACCESS_GETINSTANCERIGHTS

Purpose

Retrieve access control list information for the requested instance <aid>/<iid>.

Parameters

aid (Type = T_LONGLONG)

The Id of the application element.

iid (Type = T_LONGLONG)

The Id of the instance.

Java Calling Sequence

```
ACL[] aclEntries = applElemAccess.getInstanceRights(aid,iid);
```

Returns:

Return-Name:	aclEntries
Return-Type:	ACLSequence

The access control list entries of the given instance.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLELEMACCESS_GETINSTANCES

Purpose

Perform the Query.

The number of different application elements which are requested are exactly defined by the definition of the query given in the field anuSeq of the QueryStructure. The number of attributes for each application element is also given in the definition. The number of matching instances (their attributes) is not defined by the query and can be a large amount. Therefore only one iterator for the attribute values is defined.

Parameters

aoq (Type = QueryStructure)

The query definition.

how_many (Type = T_LONG)

Maximum number of instances in each result set. Valid arguments are:

how_many = 0 : report all instances found

how_many > 0 : report at most the given number of instances found

Java Calling Sequence

ElemResultSet[] elemResultSet = applElemAccess.getInstances(aoq,how_many);

Returns:

Return-Name:	elemResultSet
Return-Type:	ElemResultSetSequence

The result set with the requested attribute values.

Examples:

Language: Java

```
import org.asam.ods.AoException;
import org.asam.ods.AoFactory;
import org.asam.ods.AoSession;
import org.asam.ods.ApplicationAttribute;
import org.asam.ods.ApplicationElement;
import org.asam.ods.ApplicationStructure;
import org.asam.ods.ApplElemAccess;
import org.asam.ods.QueryStructure;
import org.asam.ods.AIDName;
import org.asam.ods.AIDNameUnitId;
import org.asam.ods.AIDNameValueUnitId;
import org.asam.ods.ElemId;
import org.asam.ods.ElemResultSet;
import org.asam.ods.SelOpcode;
import org.asam.ods.SelOperator;
import org.asam.ods.SelValue;
import org.asam.ods.TS_Value;
```

```
    import org.asam.ods.TS_Union;
    import org.asam.ods.T_LONGLONG;


    public class UsingApplElemAccess {

       public static void showElemResultSet (ElemResultSet elemRes[]) {
          for (int i = 0; i < elemRes.length; i++) {
             System.out.println("aid = " + elemRes[i].aid.low);
             for (int j = 0; j < elemRes[i].attrValues.length; j++) {

System.out.println(elemRes[i].attrValues[j].attrValues.valName);
                // Print the values.
             }
          }
       }

       // Java application entry.
       public static void main(String[] args) {

          try {

             // Establish a session to the service (without session
                options).
             AoSession aoSession = aoFactory.newSession("");

             // Get the application model.
             ApplicationStructure as =
              aoSession.getApplicationStructure();

             // Get the appl elem access object
             ApplElemAccess aea = aoSession.getApplElemAccess();

             // Create the query structure for the method getInstances
             QueryStructure aoq;

             // The application attribute.
             ApplicationAttribute aaObj;

             // Result of the request
             ElemResultSet elemRes[];

             // Selectvalue of the Id
             T_LONGLONG iid;

             // Query on the application element AoTest
```

```
ApplicationElement ae[] = as.getElementsByBaseType (
 "AoTest");
ApplicationElement aeObj = null;

if (ae.length > 0)
{
   aeObj = ae[0];

   /* Report attribute Id, Name of the instances of
      AoTest with
    * id < 10 and name like "ZYK*" and reference to
      AoSubTest with Id 1"
    */
   aoq = new QueryStructure();
   aaObj = aeObj.getAttributeByBaseName("id");
   aoq.anuSeq = new AIDNameUnitId [2];
   aoq.anuSeq[0] = new AIDNameUnitId ();
   aoq.anuSeq[0].attr = new AIDName();
   // set Id of element
   aoq.anuSeq[0].attr.aid = aeObj.getId();
   // Set Name of attribute
   aoq.anuSeq[0].attr.aaName = aaObj.getName();

   // Build the query, two select values.
   aoq.condSeq = new SelValue[2];

   // First select on the Id
   aaObj = aeObj.getAttributeByBaseName("id");
   aoq.condSeq[0] = new SelValue();
   aoq.condSeq[0].attr = new AIDNameValueUnitId ();
   aoq.condSeq[0].attr.attr = new AIDName();
   aoq.condSeq[0].attr.attr.aid = aeObj.getId();
   aoq.condSeq[0].attr.attr.aaName = aaObj.getName();
   aoq.condSeq[0].value = new TS_Value();
   aoq.condSeq[0].value.u = new TS_Union();
   iid = new T_LONGLONG();
   iid.high = 0;
   iid.low = 10;
   aoq.condSeq[0].value.u.longlongVal(iid);
   aoq.condSeq[0].oper = SelOpcode.LT;

   // Second select on the Name
   aaObj = aeObj.getAttributeByBaseName("name");
   aoq.anuSeq[1] = new AIDNameUnitId ();
   aoq.anuSeq[1].attr = new AIDName();
   // set Id of element
```

```
      aoq.anuSeq[1].attr.aid = aeObj.getId();
      // Set name of attribute
      aoq.anuSeq[1].attr.aaName = aaObj.getName();

      // Build the query.
      aoq.condSeq[1] = new SelValue();
      aoq.condSeq[1].attr = new AIDNameValueUnitId ();
      aoq.condSeq[1].attr.attr = new AIDName();
      aoq.condSeq[1].attr.attr.aid = aeObj.getId();
      aoq.condSeq[1].attr.attr.aaName = aaObj.getName();
      aoq.condSeq[1].value = new TS_Value();
      aoq.condSeq[1].value.u = new TS_Union();
      aoq.condSeq[1].value.u.stringVal("ZYK*");
      aoq.condSeq[1].oper = SelOpcode.EQ;

      // Set the operator.
      aoq.operSeq = new SelOperator[1];
      aoq.operSeq[0] = SelOperator.OR;

      // Set the reference selection
      ae = as.getElementsByBaseType ("AoSubTest");
      ApplicationElement aeSubObj = null;

      if (ae.length > 0)
      {
         aeSubObj = ae[0];

         aoq.relInst = new ElemId();
         aoq.relInst.aid = aeSubObj.getId();
         aoq.relInst.iid = new T_LONGLONG();
         aoq.relInst.iid.low = 1;
      }

      System.out.println("Start call getInstances");
      System.out.println("attribute Id of the instances of
        AoTest with id < 10 or name like \"ZYK*\" and
        reference to AoSubTest with Id 1");
      elemRes = aea.getInstances(aoq, 100);

      showElemResultSet(elemRes);
   }

   aea = null;
   as = null;

   // Close the active session.
```

```
                aoSession.close();

             } catch (AoException aoException) {
                System.err.println("\nUsingApplElemAccess, " +
                                   aoException.toString() + ":" +
                                   "\n   " + aoException.reason);
             }

             // Exit the application.
             System.exit(0);
          }
       }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLELEMACCESS_GETINSTANCESEXT

Purpose

Perform the Query. This method can be used for a more powerful query compared to the method getInstances of this interface, with join definitions and aggregate functions.

The number of different application elements which are requested are exactly defined by the definition of the query given in the field anuSeq of the QueryStructureExt. The number of attributes for each application element is also given in the definition. The number of matching instances (their attributes) is not defined by the query and can be a large amount. Therefore only one iterator for the attribute values is defined.

Parameters

aoq (Type = QueryStructureExt)

The query definition.

how_many (Type = T_LONG)

Maximum number of instances in each result set. Valid arguments are:

how_many = 0 : report all instances found

how_many > 0 : report at most the given number of instances found

Java Calling Sequence

```
ResultSetExt[] resultSet = applElemAccess.getInstancesExt(aoq,how_many);
```

Returns:

Return-Name:	resultSet
Return-Type:	ResultSetExtSequence

The result set with the requested attribute values.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLELEMACCESS_GETRELINST

Purpose

Get related instances (Id). This method returns a sequence of related instances.

The relation name specifies the relation given in the ApplStructValue. The aid of the ElemId and the relName define the unique relation and the target application element.

Parameters

elem (Type = ElemId)

Original instance. At the RPC-API this information was stored in the field elemId of the structure GetInstRefReq and the request AOP_GetInstRef.

relName (Type = Name)

The relation name. At the RPC-API this information was stored in the field refName of the structure GetInstRefReq and the request AOP_GetInstRef.

Java Calling Sequence

```
T_LONGLONG[] instIds = applElemAccess.getRelInst(elem,relName);
```

Returns:

Return-Name:	instIds
Return-Type:	S_LONGLONG

The list of the Id of the related instances.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLELEMACCESS_GETVALUEMATRIX

Purpose

Get the value matrix of a measurement or a submatrix. If the value matrix will be built up with special submatrix link, use the interface Measurement.

Parameters

elem (Type = ElemId)

The element id. The aid has to be the appliction element Id of the measurement or submatrix.

Java Calling Sequence

```
ValueMatrix vm = applElemAccess.getValueMatrix(elem);
```

Returns:

Return-Name:	vm
Return-Type:	ValueMatrix

The value matrix

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_INVALID_BASETYPE

APPLELEMACCESS_INSERTINSTANCES

Purpose

Create instance elements of an application element. The application element is specified by the AID of the input structure. The attribute names are specified by the name of the input structure.The values of one instance element are specified in the valueseq of the input structure. You can create several instance elements in one call by filling the valueseq of the input structure. The same index in the valueseq corresponds to the attribute values of one instance element.This method returns a sequence of Id's, the order is related to the order of instance specified in the input. In case of inheritance, the method supports only instances of the same subtype per call. The returned Id's are the Id's of the related supertype instances.

The client must supply all mandatory attributes and references within one single method call; otherwise the object cannot be made persistent by the server in the database without the risk of violating any database constraint.

Parameters

val (Type = AIDNameValueSeqUnitIdSequence)

The sequence of attributes and their values. At the RPC-API this information is stored in the fields elemId and nvSeq of the structure PutValReq and the request AOP_PutValReq.

Java Calling Sequence

```
ElemIdSequence elemIds = applElemAccess.insertInstances(val);
```

Returns:

Return-Name:	elemIds
Return-Type:	ElemIdSequence

List with the Ids of the newly created instances.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLELEMACCESS_SETATTRIBUTERIGHTS

Purpose

Set the ACL information on some application element-attribute defined by <aid> and <attr_name>. The <usergroup_id> defines the usergroup the rights should be set for. <rights> defines the rights to set or to clear. If the parameter <set> is set to 'set', the rights in <rights> are set, all others are cleared. If the parameter <set> is set to 'add', the rights in <rights> are added to the existing rights. If the parameter <set> is set to 'remove', the rights in <rights> are removed from the existing rights. Restriction for the model: only one application element of the type AoUserGroup is allowed.

Parameters

aid (Type = T_LONGLONG)

The Id of the application element.

attrName (Type = T_STRING)

The name of the attribute.

usergroupId (Type = T_LONGLONG)

The usergroup to set the rights for.

rights (Type = T_LONG)

The new right for the usergroup.

set (Type = RightsSet)

What to do with the new right.

Java Calling Sequence

```
applElemAccess.setAttributeRights(aid,attrName,usergroupId,rights,set);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLELEMACCESS_SETELEMENTINITIALRIGHTS

Purpose

Set the access control list information for the initial rights on some application element defined by <aid>. The <usergroup_id> defines the usergroup the rights should be set for. <rights> defines the rights to set or to clear. If the parameter <set> is set to 'set', the rights in <rights> are set, all others are cleared. If the parameter <set> is set to 'add', the rights in <rights> are added to the existing rights. If the parameter <set> is set to 'remove', the rights in <rights> are removed from the existing rights. Restriction for the model: only one application element of the type AoUserGroup is allowed.

Parameters

aid (Type = T_LONGLONG)

The Id of the application element.

usergroupId (Type = T_LONGLONG)

The usergroup to set the initial rights for.

rights (Type = T_LONG)

The new initial rights for the usergroup. The rights constants are defined in the interface SecurityRights. The interface definition language IDL does not allow to set the values of enumerations thus the constant definitions had to be done in an interface.

refAid (Type = T_LONGLONG)

The Id of referencing application element for which the initial rights will be used. If no refAid is set the initial rights will be used for each new instance element independent of the application element.

set (Type = RightsSet)

What to do with the new initial rights.

Java Calling Sequence

```
applElemAccess.setElementInitialRights(aid,usergroupId,rights,refAid,set);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLELEMACCESS_SETELEMENTRIGHTS

Purpose

Set the ACL information on some application element defined by <aid>. The <usergroup_id> defines the usergroup the rights should be set for. <rights> defines the rights to set or to clear. If the parameter <set> is set to 'set', the rights in <rights> are set, all others are cleared. If the parameter <set> is set to 'add', the rights in <rights> are added to the existing rights. If the parameter <set> is set to 'remove', the rights in <rights> are removed from the existing rights. Restriction for the model: only one application element of the type AoUserGroup is allowed.

Parameters

aid (Type = T_LONGLONG)

The Id of the application element.

usergroupId (Type = T_LONGLONG)

The usergroup to set the rights for.

rights (Type = T_LONG)

The new rights for the usergroup. The rights constants are defined in the interface SecurityRights. The interface definition language IDL does not allow to set the values of enumerations thus the constant definitions had to be done in an interface.

set (Type = RightsSet)

What to do with the new right.

Java Calling Sequence

```
applElemAccess.setElementRights(aid,usergroupId,rights,set);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLELEMACCESS_SETINITIALRIGHTREFERENCE

Purpose

Set the flag <set> in svcattr, if this reference will be used (or not) to retrieve the Initial Rights. If more than one reference is set to true, the union (or-function) of all rights are used.

Parameters

aid (Type = T_LONGLONG)

The application element Id.

refName (Type = T_STRING)

The name of the reference.

set (Type = RightsSet)

What to do with the given reference.

Java Calling Sequence

```
applElemAccess.setInitialRightReference(aid,refName,set);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLELEMACCESS_SETINSTANCEINITIALRIGHTS

Purpose

Set the access control list information for the initial rights on some instance defined by the application element id <aid> and a sequence of instance defined by the id <iid>. The <usergroup_id> defines the usergroup the rights should be set for. <rights> defines the rights to set or to clear. If the parameter <set> is set to 'set', the rights in <rights> are set, all others are cleared. If the parameter <set> is set to 'add', the rights in <rights> are added to the existing rights. If the parameter <set> is set to 'remove', the rights in <rights> are removed from the existing rights. Restriction for the model: only one application element of the type AoUserGroup is allowed.

Parameters

aid (Type = T_LONGLONG)

The Id of the application element.

instIds (Type = S_LONGLONG)

The sequence of instance Id's.

usergroupId (Type = T_LONGLONG)

The usergroup to set the initial rights for.

rights (Type = T_LONG)

The new initial right for the usergroup.

refAid (Type = T_LONGLONG)

The Id of referencing application element for which the initial rights will be used. If no refAid is set the initial rights will be used for each new instance element independent of the application element.

set (Type = RightsSet)

Specifies what to do with the new initial rights

Java Calling Sequence

```
applElemAccess.setInstanceInitialRights(aid,instIds,usergroupId,rights,refAid,set);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLELEMACCESS_SETINSTANCERIGHTS

Purpose

Set the ACL information on some instance defined by the application element id <aid> and a sequence of instance defined by the id <iid>. The <usergroup_id> defines the usergroup the rights should be set for. <rights> defines the rights to set or to clear. If the parameter <set> is set to 'set', the rights in <rights> are set, all others are cleared. If the parameter <set> is set to 'add', the rights in <rights> are added to the existing rights. If the parameter <set> is set to 'remove', the rights in <rights> are removed from the existing rights. Restriction for the model: only one application element of the type AoUserGroup is allowed.

Parameters

aid (Type = T_LONGLONG)

The Id of the application element.

instIds (Type = S_LONGLONG)

The sequence of instance Id's.

usergroupId (Type = T_LONGLONG)

The usergroup to set the rights for.

rights (Type = T_LONG)

The new right for the usergroup.

set (Type = RightsSet)

What to do with the new right.

Java Calling Sequence

```
applElemAccess.setInstanceRights(aid,instIds,usergroupId,rights,set);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLELEMACCESS_SETRELINST

Purpose

Set the instance reference.

Parameters

elem (Type = ElemId)

The instance to add the related instances. At the RPC-API this information was stored in the field elemId1 of the structure SetInstRefReq and the request AOP_SetInstRef.

relName (Type = Name)

The name of relation. At the RPC-API this information was stored in the field refName of the structure SetInstRefReq and the request AOP_SetInstRef.

instIds (Type = S_LONGLONG)

Sequence of instance id's. At the RPC-API this information was stored in the field elemId2 of the structure SetInstRefReq and the request AOP_SetInstRef. It was not possiable to set more then one relation.

type (Type = SetType)

The type of the modification, insert, update or remove. At the RPC-API this information was stored in the field onoff of the structure SetInstRefReq and the request AOP_SetInstRef.

Java Calling Sequence

```
applElemAccess.setRelInst(elem,relName,instIds,type);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLELEMACCESS_UPDATEINSTANCES

Purpose

Update the one or more attributes of one or more instance elements. It is necessary that the input structure includes also the Id attribute; it will be used to select the instance elements. In case of inherited application elements the supertype Id has to be included. The values of one instance element are specified in the valueseq of the input structure. The same index in the valueseq corresponds to the attributes values of one instance element.

Parameters

val (Type = AIDNameValueSeqUnitIdSequence)

The sequence of attributes and their values. At least one of the attribute values sequence must be a sequence with the Id. At the RPC-API this information was stored in the fields elemId and nvSeq of the structure PutValReq and the request AOP_PutValReq.

Java Calling Sequence

```
applElemAccess.updateInstances(val);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

10.2.4 APPLICATIONATTRIBUTE

APPLICATIONATTRIBUTE_GETAPPLICATIONELEMENT

Purpose

Return the application element to which the attribute belongs.

Parameters

None.

Java Calling Sequence

ApplicationElement applElem = applAttr.getApplicationElement();

Returns:

Return-Name:	applElem
Return-Type:	ApplicationElement

The application element of the attribute.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONATTRIBUTE_GETBASEATTRIBUTE

Purpose

Get the base attribute of the application attribute.

Parameters

None.

Java Calling Sequence

BaseAttribute baseAttr = applAttr.getBaseAttribute();

Returns:

Return-Name:	baseAttr
Return-Type:	BaseAttribute

The base attribute of the application attribute. A 'null' is returned if the application attribute has no base attribute.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONATTRIBUTE_GETDATATYPE

Purpose

Get the data type of the application attribute.

Parameters

None.

Java Calling Sequence

```
DataType aaDataType = applAttr.getDataType();
```

Returns:

Return-Name:	aaDataType
Return-Type:	DataType

The data type of the application attribute.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONATTRIBUTE_GETENUMERATIONDEFINITION

Purpose

Get the definition of the enumeration.

Parameters

None.

Java Calling Sequence

```
EnumerationDefinition enumDef;
enumDef = applAttr.getEnumerationDefinition();
```

Returns:

Return-Name:	enumDef
Return-Type:	EnumerationDefinition

The ASAM ODS enumeration.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLICATIONATTRIBUTE_GETLENGTH

Purpose

Get the maximum allowed length of the value of the application attribute.

Parameters

None.

Java Calling Sequence

```
T_LONG aaLength = applAttr.getLength();
```

Returns:

Return-Name:	aaLength
Return-Type:	T_LONG

The maximum allowed length of the application attribute.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONATTRIBUTE_GETNAME

Purpose

Get the name of the application attribute.

Parameters

None.

Java Calling Sequence

```
Name aaName = applAttr.getName();
```

Returns:

Return-Name:	aaName
Return-Type:	Name

The name of the application attribute.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONATTRIBUTE_GETRIGHTS

Purpose

Retrieve access control list information of the given object.

Parameters

None.

Java Calling Sequence

```
ACL[] aclEntries = applAttr.getRights();
```

Returns:

Return-Name:	aclEntries
Return-Type:	ACLSequence

The access control list entries of the given application element.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLICATIONATTRIBUTE_GETUNIT

Purpose

Get the unit Id of the application attribute. The unit Id is only valid for the current server.

Parameters

None.

Java Calling Sequence

```
T_LONGLONG aaUnit = applAttr.getUnit();
```

Returns:

Return-Name:	aaUnit
Return-Type:	T_LONGLONG

The unit Id of the application attribute.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONATTRIBUTE_HASUNIT

Purpose

Has the attribute an unit. If this flag is set, all the attributes of the instances derived from this attribute will have an unit.

Parameters

None.

Java Calling Sequence

```
T_BOOLEAN hasUnit = aaObj.hasUnit();
```

Returns:

Return-Name:	hasUnit
Return-Type:	T_BOOLEAN

The flag if the attribute has an unit.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLICATIONATTRIBUTE_HASVALUEFLAG

Purpose

Has the attribute a value flag. If this flag is set, all the attributes of the instances derived from this attribute will have a value flag. If this flag is not set the flag in the TS_Value structure can be ignored.

Parameters

None.

Java Calling Sequence

```
T_BOOLEAN hasValueFlag = aaObj.hasValueFlag();
```

Returns:

Return-Name:	hasValueFlag
Return-Type:	T_BOOLEAN

The flag if the attribute has a value flag.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLICATIONATTRIBUTE_ISAUTOGENERATED

Purpose

Get the autogenerate flag of the application attribute.

Parameters

None.

Java Calling Sequence

T_BOOLEAN isAutogenerated = applAttr.IsAutogenerated();

Returns:

Return-Name:	isAutogenerated
Return-Type:	T_BOOLEAN

The autogenerate flag of the application attribute.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONATTRIBUTE_ISOBLIGATORY

Purpose

Get the obligatory flag of the application attribute.

Parameters

None.

Java Calling Sequence

T_BOOLEAN aaIsObligatory = applAttr.isObligatory();

Returns:

Return-Name:	aaIsObligatory
Return-Type:	T_BOOLEAN

The obligatory flag of the application attribute.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONATTRIBUTE_ISUNIQUE

Purpose

Get the unique flag of the application attribute.

Parameters

None.

Java Calling Sequence

```
T_BOOLEAN aaIsUnique = applAttr.isUnique();
```

Returns:

Return-Name:	aaIsUnique
Return-Type:	T_BOOLEAN

The unique flag of the application attribute.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONATTRIBUTE_SETBASEATTRIBUTE

Purpose

Set the base attribute of the application attribute. This allows the client to declare the application attribute (new or existing) additional to a base attribute. The application attribute will become the derived attribute of the given base attribute.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The base attribute must be unique within the application element otherwise the exception AO_DUPLICATE_BASE_ATTRIBUTE is thrown.

For performance and flexibility reasons this set-method should be used before the new application attribute is committed the first time.

If this method is called before the first commit it will not throw the following exceptions:

AO_INVALID_DATATYPE

AO_MISSING_VALUE

AO_NOT_UNIQUE.

After the first commit, there may be instances of the application attribute. These instances may cause the following problems:

AO_INVALID_DATATYPE: The datatype of the base attribute is not the same as the datatype of the instanciated attributes.

AO_MISSING_VALUE: The obligatory flag of the base attribute is set but there are one or more empty values in the instances.

AO_NOT_UNIQUE: The unique flag of the base attribute is set but the values of the instances are not unique.

The length, the name and the unit of the application attribute are not affected by this call.

Parameters

baseAttr (Type = BaseAttribute)

The base attribute.

Java Calling Sequence

```
applAttr.setBaseAttribute(baseAttr);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_DUPLICATE_BASE_ATTRIBUTE

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_DATATYPE

AO_MISSING_VALUE

AO_NOT_IMPLEMENTED

AO_NOT_UNIQUE

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONATTRIBUTE_SETDATATYPE

Purpose

Set the data type of the application attribute.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

It is not allowed to set the datatype of application attributes that represent base attributes. An attempt to set the datatype of such an application attribute will result in the exception AO_IS_BASE_ATTRIBUTE.

For performance and flexibility reasons this set-method should be used before the new application attribute is committed the first time.

If this method is called before the first commit it will not throw the following exception:

AO_INVALID_DATATYPE

After the first commit, there may be instances of the application attribute. These instances may cause the following problem:

AO_INVALID_DATATYPE: The datatype of the base attribute is not the same as the datatype of the instanciated attributes.

Parameters

aaDataType (Type = DataType)

The data type.

Java Calling Sequence

```
applAttr.setDataType(aaDataType);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_HAS_BASE_ATTRIBUTE

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_DATATYPE

AO_IS_BASE_ATTRIBUTE

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

ApplicationAttribute_setEnumerationDefinition

Purpose

Set the definition of the enumeration. This method modifies the application model, only the superuser can use this method.

Parameters

enumDef (Type = EnumerationDefinition)

The new enumeration definition.

Java Calling Sequence

applAttr.setEnumerationDefinition(enumDef);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

AO_ACCESS_DENIED

APPLICATIONATTRIBUTE_SETISAUTOGENERATED

Purpose

Set the autogenerate flag of the application attribute.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

For performance and flexibility reasons this set-method should be used before the new application attribute is committed the first time.

Parameters

isAutogenerated (Type = T_BOOLEAN)

The autogenerate flag.

Java Calling Sequence

```
applAttr.setIsAutogenerated(isAutogenerated);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_HAS_BASE_ATTRIBUTE

AO_IMPLEMENTATION_PROBLEM

AO_IS_BASE_ATTRIBUTE

AO_MISSING_VALUE

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONATTRIBUTE_SETISOBLIGATORY

Purpose

Set the obligatory flag of the application attribute.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

It is not allowed to set the obligatory flag of application attributes

that represent base attributes. An attempt to set the obligatory flag of such an application attribute will result in the exception AO_IS_BASE_ATTRIBUTE.

For performance and flexibility reasons this set-method should be used before the new application attribute is committed the first time.

If this method is called before the first commit it will not throw the following exception:

AO_MISSING_VALUE

After the first commit, there may be instances of the application attribute. These instances may cause the following problem:

AO_MISSING_VALUE: The obligatory flag of the base attribute is set but there are one or more empty values in the instances.

Parameters

aaIsObligatory (Type = T_BOOLEAN)

The obligatory flag.

Java Calling Sequence

applAttr.setIsObligatory(aaIsObligatory);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_HAS_BASE_ATTRIBUTE

AO_IMPLEMENTATION_PROBLEM

AO_IS_BASE_ATTRIBUTE

AO_MISSING_VALUE

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONATTRIBUTE_SETISUNIQUE

Purpose

Set the unique flag of the application attribute.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The server will check if the values of the instance attributes are unique. If this flag is set and the values of an attribute are not unique when using the method setValuean exception is thrown. If instances of the application element already exist that contain non-unique values and the flag shall be set this method throws an exception.

It is not allowed to set the unique flag of application attributes that represent base attributes. An attempt to set the unique flag of such an application attribute will result in the exception AO_IS_BASE_ATTRIBUTE.

If the unique flag is set to TRUE the obligatory flag is also set to TRUE. The previous values of both flag do not matter in this case. Setting the unique flag to FALSE does not affect the obligatory flag.

For performance and flexibility reasons this set-method should be used before the new application attribute is committed the first time.

If this method is called before the first commit it will not throw the following exception:

AO_MISSING_VALUE

AO_NOT_UNIQUE

After the first commit, there may be instances of the application attribute. These instances may cause the following problem:

AO_MISSING_VALUE: The obligatory flag of the base attribute is set but there are one or more empty values in the instances.

AO_NOT_UNIQUE: The unique flag of the base attribute is set but the values of the instances are not unique.

Parameters

aaIsUnique (Type = T_BOOLEAN)

The unique flag.

Java Calling Sequence

```
applAttr.setIsUnique(aaIsUnique);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_HAS_BASE_ATTRIBUTE

AO_IMPLEMENTATION_PROBLEM

AO_IS_BASE_ATTRIBUTE

AO_MISSING_VALUE

AO_NOT_IMPLEMENTED

AO_NOT_UNIQUE

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONATTRIBUTE_SETLENGTH

Purpose

Set the maximum allowed length of the application attribute.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

This method is useful for ODS database design tools. Negative length values are not allowed.

This method provides only a hint to a database server in the design phase which size the data entries may have. The length is ignored for all other datatypes than DT_STRING and DS_*.

For performance and flexibility reasons this set-method should be used before the new application attribute is committed the first time.

If this method is called before the first commit it will not throw the following exception:

AO_HAS_INSTANCES

After the first commit, there may be instances of the application attribute. These instances may cause the exception AO_HAS_INSTANCES if the instances of the application attribute are not empty.

Parameters

aaLength (Type = T_LONG)

The maximum attribute length.

Java Calling Sequence

```
applAttr.setLength(aaLength);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_HAS_INSTANCES

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_LENGTH

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONATTRIBUTE_SETNAME

Purpose

Set the name of an application attribute.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The name must be unique.

For performance and flexibility reasons this set-method should be used before the new application attribute is committed the first time.

The name of an application attribute must not exceed the maximum name length of the underlying physical storage. The current specification of the physical storage restricts it to 30 characters.

Parameters

aaName (Type = Name)

The application attribute name.

Java Calling Sequence

```
applAttr.setName(aaName);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_DUPLICATE_NAME

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONATTRIBUTE_SETRIGHTS

Purpose

The given usergroup the rights should be set for. <rights> defines the rights to set or to clear. If the parameter <set> is set to 'set', the rights in <rights> are set, all others are cleared. If the parameter <set> is set to 'add', the rights in <rights> are added to the existing rights. If the parameter <set> is set to 'remove', the rights in <rights> are removed from the existing rights.

Parameters

usergroup (Type = InstanceElement)

The usergroup for which the rights will be modified.

rights (Type = T_LONG)

The new right for the usergroup. The rights constants are defined in the interface SecurityRights. The interface definition language IDL does not allow to set the values of enumerations thus the constant definitions had to be done in an interface.

set (Type = RightsSet)

What to do with the new right.

Java Calling Sequence

```
applAttr.setRights(usergroup,rights,set);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLICATIONATTRIBUTE_SETUNIT

Purpose

Set the unit Id of an application attribute.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The unit Id is only valid for the current server. If instances of the application attribute exist, the respective values are automatically converted to the new unit. If there is no known conversion an exception is thrown.

The automatic conversion can be avoided if the unit is set to zero. After that the transaction must be committed. In the next step the new unit may be set in another transaction.

The automatic conversion is done only for the following datatypes:

DT_BYTE

DT_COMPLEX

DT_DCOMPLEX

DT_DOUBLE

DT_FLOAT

DT_LONG

DT_LONGLONG

DT_SHORT

as well as for the corresponding sequence datatypes. For complex datatypes the real and imaginary part are converted separately.

If the unit of an attribute is set the unit is constant. If the value of the attribute has another unit the value is calibrated to the unit of the application attribute. If there is no unit at the application attribute the unit at the attribute value is stored and reported on request at the instance.

For performance and flexibility reasons this set-method should be used before the new application attribute is committed the first time.

If this method is called before the first commit it will not throw the following exceptions:

AO_INCOMPATIBLE_UNITS

AO_MATH_ERROR

After the first commit, there may be instances of the application attribute. These instances may cause the following problems:

AO_INCOMPATIBLE_UNITS: No conversion rules is known to convert the unit.

AO_MATH_ERROR: Converting the values to the new unit results in data overflow or underflow or a division by zero is detected.

Parameters

aaUnit (Type = T_LONGLONG)

The unit Id.

Java Calling Sequence

```
applAttr.setUnit(aaUnit);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INCOMPATIBLE_UNITS

AO_MATH_ERROR

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_UNKNOWN_UNIT

APPLICATIONATTRIBUTE_WITHUNIT

Purpose

Specifies whether the attribute will have a unit or not. A call to the method setUnit() will automatically set the withUnit flag to TRUE.

Parameters

withUnit (Type = T_BOOLEAN)

The flag value: TRUE, if the attribute will have a unit, FALSE otherwise.

Java Calling Sequence

```
aaObj.withUnit(TRUE);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLICATIONATTRIBUTE_WITHVALUEFLAG

Purpose

Specifies whether the attribute will have a value flag or not. If this flag is not set the flag of the TS_Value will be ignored by the server.

Parameters

withValueFlag (Type = T_BOOLEAN)

The flag value: TRUE, if the attribute will have a value flag, FALSE otherwise.

Java Calling Sequence

```
aaObj.withValueFlag(TRUE);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

10.2.5 APPLICATIONELEMENT

APPLICATIONELEMENT_CREATEATTRIBUTE

Purpose

Create a new application attribute on the server.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The properties of the new application attribute may be changed via the set-methods of the ApplicationAttribute interface.

For performance reasons it is recommended to set all required properties of an application attribute before it is committed the first time. This avoids database cross-checks for each attribute.

The default properties of a new application attribute are:

BaseAttribute	NULL
DataType	DT_UNKNOWN
IsObligatory	0
IsUnique	0
Length	0
Name	"AUTOGEN"
Unit	NULL

If there are already instances of the application element the values of the existing instances of the new attribute are set to undefined (flag AO_VF_DEFINED is set to zero).

The exception AO_DUPLICATE_NAME name occurs if there is already another application attribute with the name "AUTOGEN".

Parameters

None.

Java Calling Sequence

```
ApplicationAttribute applAttr = applElem.createAttribute();
```

Returns:

Return-Name: applAttr

Return-Type: ApplicationAttribute

The new application attribute.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_DUPLICATE_NAME

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_CREATEINSTANCE

Purpose

Create an instance of the application element.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The instance gets permanent when the transaction is committed. All attributes connected to the application element are automatically created and connected to the instance. The values of the attributes can be set by the method setValue of the interface InstanceElement.

Parameters

ieName (Type = Name)

The instance name.

Java Calling Sequence

```
InstanceElement instElem = applElem.createInstance(ieName);
```

Returns:

Return-Name:	instElem
Return-Type:	InstanceElement

The new instance.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_CREATEINSTANCES

Purpose

Create a list with instances. The attribute are given with the name of the sequence. The values of the attributes are given in the value sequence. The index in the different value sequences match for one instance element. The index in the instance element sequence of the related instances match for the instance with the same index in the value sequence.

Parameters

attributes (Type = NameValueSeqUnitSequence)

The attributes of the new created instances.

relatedInstances (Type = ApplicationRelationInstanceElementSeqSequence)

The list with related instances for different application relations.

Java Calling Sequence

```
InstanceElement[] instElems = applElem.createInstances(attributes, relatedInstances);
```

Returns:

Return-Name:	instElems
Return-Type:	InstanceElementSequence

The new instances.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO:TRANSACTION_NOT_ACTIVE

AO_INVALID_REQUEST

APPLICATIONELEMENT_GETALLRELATEDELEMENTS

Purpose

Get a list of all related application elements connected to this application element.

Parameters

None.

Java Calling Sequence

```
ApplicationElement[] applElems = applElem.getAllRelatedElements();
```

Returns:

Return-Name:	applElems
Return-Type:	ApplicationElementSequence

The related application elements.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_GETALLRELATIONS

Purpose

Get a list of all application relations connected to this application element. The inverse relation of relations connected to other application elements pointing to the given application elements are not returned.

Parameters

None.

Java Calling Sequence

ApplicationRelation[] applRels = applElem.getAllRelations();

Returns:

Return-Name: applRels

Return-Type: ApplicationRelationSequence

The application relations of the application element.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_GETAPPLICATIONSTRUCTURE

Purpose

Get the application model to which the application element belongs. The application model returned is the same as that returned from the method getApplicationStructure of the Interface AoSession. This method guarantees that the client software is able to return to the session in case the session object is not available.

Parameters

None.

Java Calling Sequence

```
ApplicationStructure applStruct = applElem.getApplicationStructure();
```

Returns:

Return-Name:	applStruct
Return-Type:	ApplicationStructure

The application model to which the application element belongs.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLICATIONELEMENT_GETATTRIBUTEBYBASENAME

Purpose

Get the application attribute of an application element which is inherited from the base attribute with the given name. The base name of the attribute is case insensitive (the base model itself is case insensitive; ID, Id, id,... are treated as the same items) and may not contain wildcard characters.

Note: The base model is case blind, e.g. Id, ID and id is all the same base attribute.

Parameters

baName (Type = Name)

The base attribute name.

Java Calling Sequence

```
ApplicationAttribute applAttr = applElem.getAttributeByBaseName(baName);
```

Returns:

Return-Name:	applAttr
Return-Type:	ApplicationAttribute

The application attribute.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_GETATTRIBUTEBYNAME

Purpose

Get the application attribute of an application element which has the given name. The name is case sensitive and may not contain wildcard characters.
Note: The application model is (in contrary to the base model) case sensitive; eg ID, Id, and id are three different items.

Note: The application model is case sensitive, eg Id andID are different application attributes, don't use this misleading attribute name.

Parameters

aaName (Type = Name)

The application attribute name.

Java Calling Sequence

ApplicationAttribute applAttr = applElem.getAttributeByName(aaName);

Returns:

Return-Name:	applAttr
Return-Type:	ApplicationAttribute

The application attribute.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_GETATTRIBUTES

Purpose

Get a list of the application attributes of an application element. The reference attributes are not returned.

Parameters

aaPattern (Type = Pattern)

The name or the search pattern for the application attribute name.

Java Calling Sequence

ApplicationAttribute[] applAttrs = applElem.getAttributes(aaPattern);

Returns:

Return-Name:	applAttrs
Return-Type:	ApplicationAttributeSequence

The application attributes.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_GETBASEELEMENT

Purpose

Get the base element of an application element.

Parameters

None.

Java Calling Sequence

BaseElement baseElem = applElem.getBaseElement();

Returns:

Return-Name:	baseElem
Return-Type:	BaseElement

The base element.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_GETID

Purpose

Get the Id of an application element.

Parameters

None.

Java Calling Sequence

T_LONGLONG aeId = applElem.getId();

Returns:

Return-Name:	aeId
Return-Type:	T_LONGLONG

The Id of the application element.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_GETINITIALRIGHTRELATIONS

Purpose

Get all relations which are used to retrieve the instances to create the initial rights of the new created instance element. If there is more than one application relation, the initial rights of each related instance are 'ored' to the list of the initial rights.

Parameters

None.

Java Calling Sequence

ApplicationRelation[] applRels = applElem.getInitialRightRelations();

Returns:

Return-Name:	applRels
Return-Type:	ApplicationRelationSequence

The sequence with the application relations which will be used to create the initial rights of the new created instance element.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLICATIONELEMENT_GETINITIALRIGHTS

Purpose

Retrieve access control list information for the initial rights of the given object.

Parameters

None.

Java Calling Sequence

```
InitialRight[] initialRights = applElem.getInitialRights();
```

Returns:

Return-Name:	initialRights
Return-Type:	InitialRightSequence

The access control list entries with the initial rights of the given application element.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLICATIONELEMENT_GETINSTANCEBYID

Purpose

Get the instance element specified by the given Id. If the Id of the instance is not unique an exception is thrown.

Parameters

ieId (Type = T_LONGLONG)

The instance element Id.

Java Calling Sequence

```
InstanceElement instElem = applElem.getInstanceById(ieId);
```

Returns:

Return-Name:	instElem
Return-Type:	InstanceElement

The instance element.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_GETINSTANCEBYNAME

Purpose

Get the instance element specified by the given name. If the name of the instance is not unique an exception is thrown.

This is a convienience method for instance elements with unique names. If there are duplicate names for instance use the method getInstances instead and specify the requested name as pattern parameter.

Parameters

ieName (Type = Name)

The instance element name.

Java Calling Sequence

```
InstanceElement instElem = applElem.getInstanceByName(ieName);
```

Returns:

Return-Name:	instElem
Return-Type:	InstanceElement

The instance element.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_DUPLICATE_NAME

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_GETINSTANCES

Purpose

Get the instances whose names match the pattern. The pattern is case sensitive and may contain wildcard characters.

Parameters

iePattern (Type = Pattern)

The name or the search pattern for the instance element name.

Java Calling Sequence

InstanceElementIterator ieIterator = applElem.getInstances(iePattern);

Returns:

Return-Name:	ieIterator
Return-Type:	InstanceElementIterator

The instance elements.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_GETNAME

Purpose

Get the name of an application element.

Parameters

None.

Java Calling Sequence

Name aeName = applElem.getName();

Returns:

Return-Name:	aeName
Return-Type:	Name

The name of the application element.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

ApplicationElement_getRelatedElementsByRelationship

Purpose

Get related application elements connected via the specified relationship.

Parameters

aeRelationship (Type = Relationship)

The requested relationship.

Java Calling Sequence

```
ApplicationElement[] applElems =
applElem.getRelatedElementsByRelationship(aeRelationship);
```

Returns:

Return-Name:	applElems
Return-Type:	ApplicationElementSequence

The related application elements.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_RELATIONSHIP

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_GETRELATIONSBYTYPE

Purpose

Get application relations of the requested type connected from this application element. The inverse relations are not returned.

Parameters

aeRelationType (Type = RelationType)

The requested relation type.

Java Calling Sequence

```
ApplicationRelation[] applRels = applElem.getRelationsByType(aeRelationType);
```

Returns:

Return-Name:	applRels
Return-Type:	ApplicationRelationSequence

The application relations.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_RELATION_TYPE

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_GETRIGHTS

Purpose

Retrieve access control list information of the given object.

Parameters

None.

Java Calling Sequence

```
ACL[] aclEntries = applElem.getRights();
```

Returns:

Return-Name:	aclEntries
Return-Type:	ACLSequence

The access control list entries of the given application element.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLICATIONELEMENT_GETSECURITYLEVEL

Purpose

Get the security level of the application element. The security level tells if there is a security check for both application element and instance elements or only for the application attributes, the instance elements or none at all.

Parameters

None.

Java Calling Sequence

```
SecurityLevel secLevel = applElem.getSecurityLevel();
```

Returns:

Return-Name:	secLevel
Return-Type:	T_LONG

The current security level. The security level constants are defined in the interface SecurityLevel. The interface definition language IDL does not allow to set the values of enumerations thus the constant definitions had to be done in an interface.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLICATIONELEMENT_LISTALLRELATEDELEMENTS

Purpose

Get the names of all related application elements.

Parameters

None.

Java Calling Sequence

```
Name[] applElemNames = applElem.listAllRelatedElements();
```

Returns:

Return-Name:	applElemNames
Return-Type:	NameSequence

The names of the related application elements.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_LISTATTRIBUTES

Purpose

Get the application attribute names of the application element. There are no attribute names returned in the result list that contain a reference to another application element.

Parameters

aaPattern (Type = Pattern)

The name or the search pattern for the application attribute name.

Java Calling Sequence

Name[] applAttrNames = applElem.listAttributes(aaPattern);

Returns:

Return-Name:	applAttrNames
Return-Type:	NameSequence

The names of the application attributes.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_LISTINSTANCES

Purpose

Get the names of the instances whose names match the pattern. The pattern is case sensitive and may contain wildcard characters.

Parameters

aaPattern (Type = Pattern)

The name or the search pattern for the application attribute name.

Java Calling Sequence

NameIterator ieNameIterator = applElem.listInstances(aaPattern);

Returns:

Return-Name:	ieNameIterator
Return-Type:	NameIterator

The names of the instances.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_LISTRELATEDELEMENTSBYRELATIONSHIP

Purpose

Get the names of related application elements connected via the specified relationship.

Parameters

aeRelationship (Type = Relationship)

The requested relationship.

Java Calling Sequence

```
Name[] applElemNames =
applElem.listRelatedElementsByRelationship(aeRelationship);
```

Returns:

Return-Name:	applElemNames
Return-Type:	NameSequence

The names of the related application elements.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_RELATIONSHIP

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_REMOVEATTRIBUTE

Purpose

Remove an application attribute from an application element. If there are instances of the application element the attribute of the existing instances change from application to instance attributes.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

Parameters

applAttr (Type = ApplicationAttribute)

The application attribute to remove.

Java Calling Sequence

applElem.removeAttribute(applAttr);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_FOUND

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_REMOVEINSTANCE

Purpose

Remove an instance from the application element.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The instance is removed from the server when the transaction is committed. If the recursive flag is set all children of the instance are also deleted. Removing instances is allowed only if there are no references(relations) to this instance. If the recursive flag is set a reference to one of the children is not allowed and will cause an exception.

Parameters

ieId (Type = T_LONGLONG)

The instance Id.

recursive (Type = T_BOOLEAN)

The recursive flag.

Java Calling Sequence

```
applElem.removeInstance(ieId,recursive);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_HAS_REFERENCES

AO_IMPLEMENTATION_PROBLEM

AO_NOT_FOUND

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_SETBASEELEMENT

Purpose

Set the base element of the application element.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The assignment to the current base element is overwritten. If there are instances of the application element or references to the application element an exception is thrown.

Parameters

baseElem (Type = BaseElement)

The base element.

Java Calling Sequence

applElem.setBaseElement(baseElem);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_HAS_INSTANCES

AO_HAS_REFERENCES

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_SETINITIALRIGHTRELATION

Purpose

Set for the given application element, which relation will be used to determine the initial rights for the new created instances.

Parameters

applRel (Type = ApplicationRelation)

The application relation which will be used to determine the initial rights. The relation range of the application relation must be [1:1] otherwise the server cannot find a unique instance element to retrieve the initial rights.

set (Type = T_BOOLEAN)

Set or remove the relation for the initial rights. If this parameter is true the relation will be set otherwise removed.

Java Calling Sequence

```
applElem.setInitialRightRelation(applRel,set);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLICATIONELEMENT_SETINITIALRIGHTS

Purpose

The given usergroup the initial rights should be set for. <rights> defines the rights to set or to clear. If the parameter <set> is set to 'set', the rights in <rights> are set, all others are cleared. If the parameter <set> is set to 'add', the rights in <rights> are added to the existing rights. If the parameter <set> is set to 'remove', the rights in <rights> are removed from the existing rights.

Parameters

usergroup (Type = InstanceElement)

The usergroup for which the initial rights will be modified.

rights (Type = T_LONG)

The new initial rights for the usergroup.The rights constants are defined in the interface SecurityRights. The interface definition language IDL does not allow to set the values of enumerations thus the constant definitions had to be done in an interface.

refAid (Type = T_LONGLONG)

The Id of referencing application element for which the initial rights will be used. If no refAid is set the initial rights will be used for each new instance element independent of the application element.

set (Type = RightsSet)

What to do with the new initial rights.

Java Calling Sequence

```
applElem.setInitialRights(usergroup,rights,refAid,set);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLICATIONELEMENT_SETNAME

Purpose

Set the name of the application element.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The name of the application element must be unique.

The name of an application element must not exceed the maximum name length of the underlying physical storage. The current specification of the physical storage restricts it to 30 characters.

Parameters

aeName (Type = Name)

The application element name.

Java Calling Sequence

applElem.setName(aeName);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_DUPLICATE_NAME

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONELEMENT_SETRIGHTS

Purpose

The given usergroup the rights should be set for. <rights> defines the rights to set or to clear. If the parameter <set> is set to 'set', the rights in <rights> are set, all others are cleared. If the parameter <set> is set to 'add', the rights in <rights> are added to the existing rights. If the parameter <set> is set to 'remove', the rights in <rights> are removed from the existing rights.

Parameters

usergroup (Type = InstanceElement)

The usergroup for which the rights will be modified.

rights (Type = T_LONG)

The new right for the usergroup. The rights constants are defined in the interface SecurityRights. The interface definition language IDL does not allow to set the values of enumerations thus the constant definitions had to be done in an interface.

set (Type = RightsSet)

What to do with the new right.

Java Calling Sequence

```
applElem.setRights(usergroup,rights,set);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLICATIONELEMENT_SETSECURITYLEVEL

Purpose

Set the security level for the given application element. If the security level is added the client is responsable for the access control list entries of the existing objects.

Parameters

secLevel (Type = T_LONG)

The new security level.The security level constants are defined in the interface SecurityLevel. The interface definition language IDL does not allow to set the values of enumerations thus the constant definitions had to be done in an interface.

set (Type = RightsSet)

What to do with the new security level.

Java Calling Sequence

applElem.setSecurityLevel(secLevel,set);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

10.2.6 ApplicationRelation

ApplicationRelation_getBaseRelation

Purpose

Get the base relation of the application relation.

Parameters

None.

Java Calling Sequence

```
BaseRelation baseRel = applRel.getBaseRelation();
```

Returns:

Return-Name:	baseRel
Return-Type:	BaseRelation

The base relation of the application relation. A 'null' is returned if the application relation has no base relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONRELATION_GETELEM1

Purpose

Get the first application element of the application relation.

Parameters

None.

Java Calling Sequence

```
ApplicationElement applElem = applRel.getElem1();
```

Returns:

Return-Name:	applElem
Return-Type:	ApplicationElement

The first application element of the application relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONRELATION_GETELEM2

Purpose

Get the second application element of the application relation.

Parameters

None.

Java Calling Sequence

```
ApplicationElement applElem = applRel.getElem2();
```

Returns:

Return-Name:	applElem
Return-Type:	ApplicationElement

The second application element of the application relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONRELATION_GETINVERSERELATIONNAME

Purpose

Get the inverse name of the application relation. The inverse name of an application relation is the name of the relation seen from the other application element.

Parameters

None.

Java Calling Sequence

```
Name arInvName = applRel.getInverseRelationName();
```

Returns:

Return-Name:	arInvName
Return-Type:	Name

The inverse name of the application relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONRELATION_GETINVERSERELATIONRANGE

Purpose

Get the inverse relation range of the application relation.

Parameters

None.

Java Calling Sequence

```
RelationRange arRange = applRel.getInverseRelationRange();
```

Returns:

Return-Name:	arRange
Return-Type:	RelationRange

The inverse relation range of the application relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONRELATION_GETINVERSERELATIONSHIP

Purpose

Get the inverse relationship of the application relation.

Parameters

None.

Java Calling Sequence

```
Relationship relationship = applRel.getInverseRelationship();
```

Returns:

Return-Name:	relationship
Return-Type:	Relationship

The inverse relationship of the application relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONRELATION_GETRELATIONNAME

Purpose

Get the name of the application relation.

Parameters

None.

Java Calling Sequence

```
Name arName = applRel.getRelationName();
```

Returns:

Return-Name:	arName
Return-Type:	Name

The name of the application relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONRELATION_GETRELATIONRANGE

Purpose

Get the relation range of the application relation.

Parameters

None.

Java Calling Sequence

RelationRange arRange = applRel.getRelationRange();

Returns:

Return-Name:	arRange
Return-Type:	RelationRange

The relation range of the application relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONRELATION_GETRELATIONSHIP

Purpose

Get the relationship of the application relation.

Parameters

None.

Java Calling Sequence

```
Relationship relationship = applRel.getRelationship();
```

Returns:

Return-Name:	relationship
Return-Type:	Relationship

The relationship of the application relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONRELATION_GETRELATIONTYPE

Purpose

Get the relation type of the application relation.

Parameters

None.

Java Calling Sequence

```
RelationType arType = applRel.getRelationType();
```

Returns:

Return-Name:	arType
Return-Type:	RelationType

The relation type of the application relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONRELATION_SETBASERELATION

Purpose

Set the base relation of the application relation.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The relation type and relation range is copied from the base relation. The previous values get lost.

Parameters

baseRel (Type = BaseRelation)

The base relation.

Java Calling Sequence

applRel.setBaseRelation(baseRel);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_RELATION

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONRELATION_SETELEM1

Purpose

Set the first application element of the application relation.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

Parameters

applElem (Type = ApplicationElement)

The application element.

Java Calling Sequence

applRel.setElem1(applElem);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_ELEMENT

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONRELATION_SETELEM2

Purpose

Set the second application element of the application relation.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

Parameters

applElem (Type = ApplicationElement)

The application element.

Java Calling Sequence

```
applRel.setElem2(applElem);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_ELEMENT

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONRELATION_SETINVERSERELATIONNAME

Purpose

Set the name of an application relation.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

Parameters

arInvName (Type = Name)

The inverse application relation name.

Java Calling Sequence

applRel.setInverseRelationName(arInvName);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONRELATION_SETINVERSERELATIONRANGE

Purpose

Set the relation range of an application relation.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

It is only allowed to set the relation type if no base relation is defined.

Parameters

arRelationRange (Type = RelationRange)

The inverse relation range.

Java Calling Sequence

applRel.setInverseRelationRange(arRelationRange);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_HAS_BASE_RELATION

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_RELATION_RANGE

AO_IS_BASE_RELATION

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONRELATION_SETRELATIONNAME

Purpose

Set the name of an application relation.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The name of an application relation must not exceed the maximum name length of the underlying physical storage. The current specification of the physical storage restricts it to 30 characters.

Parameters

arName (Type = Name)

The application relation name.

Java Calling Sequence

applRel.setRelationName(arName);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

ApplicationRelation_setRelationRange

Purpose

Set the relation range of an application relation.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

It is only allowed to set the relation type if no base relation is defined.

Parameters

arRelationRange (Type = RelationRange)

The relation range.

Java Calling Sequence

```
applRel.setRelationRange(arRelationRange);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_HAS_BASE_RELATION

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_RELATION_RANGE

AO_IS_BASE_RELATION

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONRELATION_SETRELATIONTYPE

Purpose

Set the relation type of an application relation.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The relationship is automatically set when the relation type is set. It is only allowed to set the relation type if no base relation is defined.

Parameters

arRelationType (Type = RelationType)

The relation type.

Java Calling Sequence

```
applRel.setRelationType(arRelationType);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_HAS_BASE_RELATION

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_RELATION_TYPE

AO_IS_BASE_RELATION

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.7 APPLICATIONSTRUCTURE

ApplicationStructure is the interface that allows to access the ASAM ODS application model which currently is used by the server (please note the difference in the wording between structure and model). This is similar to BaseStructure versus ASAM ODS base model.

APPLICATIONSTRUCTURE_CHECK

Purpose

Check the application model for ASAM ODS conformity. The first error found is reported by an exception. The following checks are performed:

- Each application element must be derived from a valid base element.
- An application attribute may be derived from a base attribute. It is not allowed within one application element to derive more than one application attribute from the same base attribute. It is allowed that application attributes are not derived from any base attribute.
- All application elements must have at least the mandatory attributes.
- Each application elements must be identified by a unique Asam path. No "floating" application elements are allowed.
- All relations required by the base model must be present.

Parameters

None.

Java Calling Sequence

```
applStruct.check();
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_DUPLICATE_BASE_ATTRIBUTE

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_RELATION

AO_MISSING_ATTRIBUTE

AO_MISSING_RELATION

AO_MISSING_APPLICATION_ELEMENT

AO_NOT_IMPLEMENTED

AO_NO_PATH_TO_ELEMENT

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONSTRUCTURE_CREATEELEMENT

Purpose

Create a new application element in the application model.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The information whether or not the new application element is a top level element is taken from the specified base element. The Id of the application element is set automatically. The mandatory base attributes are created automatically. Optional attributes have to be created by the calling program. The application attribute interface methods may be used to modify the attributes.

Parameters

baseElem (Type = BaseElement)

The base element from which the application element is derived.

Java Calling Sequence

ApplicationElement applElem = applStruct.createElement(baseElem);

Returns:

Return-Name:	applElem
Return-Type:	ApplicationElement

The new application element.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.ApplicationStructure;
  import org.asam.ods.ApplicationElement;
  import org.asam.ods.BaseStructure;
  import org.asam.ods.BaseElement;
  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");
  // Session successfully created ?
  if (session != null) {
     // Get application model.
       ApplicationStructure as =
        session.getApplicationStructure();
     // Get base model.
       BaseStructure bs = session.getBaseStructure();
     // Get a base element by type.
     BaseElement be = bs.getElementByType("AoSubmatrix");
     // Start the transaction
     session.startTransaction();
     // Create a new application element.
```

```
        ApplicationElement ae = as.createElement(be);
      ...
        // Commit the transaction
        session.commitTransaction();
          // Close the session.
        session.close();
      }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONSTRUCTURE_CREATEENUMERATIONDEFINITION

Purpose

Create a new enumeration definition. This method modifies the application model and is only allowed for the superuser.

Parameters

enumName (Type = T_STRING)

Name of the enumeration

Java Calling Sequence

```
EnumerationDefinition newEnum;
newEnum = applStruct.createEnumerationDefinition(enumName);
```

Returns:

Return-Name:	newEnum
Return-Type:	EnumerationDefinition

The new created enumeration

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

AO_ACCESS_DENIED

APPLICATIONSTRUCTURE_CREATEINSTANCERELATIONS

Purpose

Create the relation between a list of instances. The number of instances in both list must be identical. The application element of the instances in each list must be identical. The application elements must match the application elements of the application relation. The index in the list of the instances defines related instances.

Parameters

applRel (Type = ApplicationRelation)

The application relation.

elemList1 (Type = InstanceElementSequence)

The list with the instances of one application element for which the relation will be created.

elemList2 (Type = InstanceElementSequence)

The list with the related instances.

Java Calling Sequence

```
applStruct.createInstanceRelations(applRel, elemList1, elemList2);
```

Returns:

none

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

AO_INVALID_REQUEST

APPLICATIONSTRUCTURE_CREATERELATION

Purpose

Create a new relation.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The relation is part of the application model. The application relation interface methods may be used to modify the relation.

The default properties of a new application relation are:

BaseRelation	NULL
Element1	NULL
Element2	NULL
Range	-2, -2
Name	NULL
Type	INFO

When element 1 or element 2 is set before the name of the relation is specified, the name of the application relation is set to "AUTOGEN".

Parameters

None.

Java Calling Sequence

ApplicationRelation applRel = applStruct.createRelation();

Returns:

Return-Name:	applRel
Return-Type:	ApplicationRelation

The new application relation.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.ApplicationStructure;
  import org.asam.ods.ApplicationRelation;

  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");

  // Session successfully created ?
  if (session != null) {
         session.StartTransaction();
```

```
        // Get application model.
           ApplicationStructure as =
          session.getApplicationStructure();

    // Create a new application relation.
    ApplicationRelation ar = as.createRelation();

    ...
    session.commitTransaction();
    // Close the session.
    session.close();

}
```

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONSTRUCTURE_GETELEMENTBYID

Purpose

Get the application element with the requested Id.

Parameters

aeId (Type = T_LONGLONG)

The Id of the requested application element.

Java Calling Sequence

ApplicationElement applElem = applStruct.getElementById(aeId);

Returns:

Return-Name:	applElem
Return-Type:	ApplicationElement

The requested application element.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.ApplicationStructure;
  import org.asam.ods.ApplicationElement;

  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");

  // Session successfully created ?
  if (session != null) {

     // Get application model.
          ApplicationStructure as =
         session.getApplicationStructure();

     // Get an application element by id.
     ApplicationElement ae = as.getElementById(69);

     ...

     // Close the session.
     session.close();

  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONSTRUCTURE_GETELEMENTBYNAME

Purpose

Get the application element with the requested name.

Parameters

aeName (Type = Name)

The name of the requested application element.

Java Calling Sequence

ApplicationElement applElem = applStruct.getElementByName(aeName);

Returns:

Return-Name:	applElem
Return-Type:	ApplicationElement

The requested application element.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.ApplicationStructure;
  import org.asam.ods.ApplicationElement;

  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");

  // Session successfully created ?
  if (session != null) {

     // Get application model.
          ApplicationStructure as =
          session.getApplicationStructure();

     // Get an application element by name.
     ApplicationElement ae = as.getElementByName("myEngine");

     ...

     // Close the session.
     session.close();

  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONSTRUCTURE_GETELEMENTS

Purpose

Get the application elements whose names match the pattern. The pattern is case sensitive and may contain wildcard characters.

Parameters

aePattern (Type = Pattern)

The name or the search pattern for the requested application elements.

Java Calling Sequence

ApplicationElement[] applElems = applStruct.getElements(aePattern);

Returns:

Return-Name:	applElems
Return-Type:	ApplicationElementSequence

The requested application elements.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.ApplicationStructure;
  import org.asam.ods.ApplicationElement;

  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");

  // Session successfully created ?
  if (session != null) {

     // Get application model.
          ApplicationStructure as =
          session.getApplicationStructure();

     // Get a list of application elements.
     ApplicationElementSeq aeSeq[] = as.getElements("*");

     ...

     // Close the session.
     session.close();

  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONSTRUCTURE_GETELEMENTSBYBASETYPE

Purpose

Get the names of application elements that are derived from the specified base element.

Parameters

aeType (Type = BaseType)

The requested base element type. The base element type can be a pattern.

Java Calling Sequence

ApplicationElement[] applElems = applStruct.getElementsByBaseType(aeType);

Returns:

Return-Name:	applElems
Return-Type:	ApplicationElementSequence

The requested application element names.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.ApplicationStructure;
  import org.asam.ods.ApplicationElement;

  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");

  // Session successfully created ?
  if (session != null) {

     // Get application model.
          ApplicationStructure as =
           session.getApplicationStructure();

     // Get a list of application elements by type.
     ApplicationElementSeq aeSeq[] =
     as.getElementsByBaseType("AoTest");

     ...

     // Close the session.
     session.close();

  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_BASETYPE

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONSTRUCTURE_GETENUMERATIONDEFINITION

Purpose

Get the specified enumeration definition.

Parameters

enumName (Type = T_STRING)

Name of the requested enumeration.

Java Calling Sequence

```
EnumerationDefinition enumDef;
enumDef = applStruct.getEnumerationDefinition(enumName);
```

Returns:

Return-Name:	enumDef
Return-Type:	EnumerationDefinition

The enumeration definition.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLICATIONSTRUCTURE_GETINSTANCEBYASAMPATH

Purpose

Get the instance element specified by the ASAM path.

Parameters

asamPath (Type = Name)

The ASAM path of the requested instance element.

Java Calling Sequence

InstanceElement instElem = applStruct.getInstanceByAsamPath(asamPath);

Returns:

Return-Name:	instElem
Return-Type:	InstanceElement

The requested instance element.

Examples:

Language: Java

```
InstanceElement ie;
  Name asamPath;
  ie = as.getInstanceByAsamPath(asamPath);
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_ASAM_PATH

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONSTRUCTURE_GETINSTANCESBYID

Purpose

Get the instance elements specified by the element id.

Parameters

ieIds (Type = ElemIdSequence)

The sequence with the element id.

Java Calling Sequence

InstanceElement[] instElems = applStruct.getInstancesById(ieIds);

Returns:

Return-Name:	instElems
Return-Type:	InstanceElementSequence

The requested instance element sequence.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_ASAM_PATH

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONSTRUCTURE_GETRELATIONS

Purpose

Returns the relations between two application elements.

Parameters

applElem1 (Type = ApplicationElement)

The first application element.

applElem2 (Type = ApplicationElement)

The second application element.

Java Calling Sequence

ApplicationRelation[] applRels = applStruct.getRelations(applElem1,applElem2);

Returns:

Return-Name:	applRels
Return-Type:	ApplicationRelationSequence

The relations between the specified application elements.

Examples:

Language: Java

```
ApplicationRelationSeq arSeq[];
        ApplicationElement ae1, ae2;
  arSeq = as.getRelations(ae1,ae2);
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONSTRUCTURE_GETSESSION

Purpose

Get the current client session in which the application model is created.

Parameters

None.

Java Calling Sequence

```
AoSession session = applStruct.getSession();
```

Returns:

Return-Name:	session
Return-Type:	AoSession

The current client session.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLICATIONSTRUCTURE_GETTOPLEVELELEMENTS

Purpose

Get the top level application elements which are inherted from the base element that matches the base type. If the given base type is no top level base element an exception is thrown. A top level application element is an application element without a father.

Parameters

aeType (Type = BaseType)

The requested base type. The base element type can be a pattern.

Java Calling Sequence

ApplicationElement[] applElems = applStruct.getTopLevelElements(aeType);

Returns:

Return-Name:	applElems
Return-Type:	ApplicationElementSequence

The top level application elements.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.ApplicationStructure;
  import org.asam.ods.ApplicationElement;

  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");

  // Session successfully created ?
  if (session != null) {

    // Get application model.
         ApplicationStructure as =
          session.getApplicationStructure();

    // Get a list of top level application elements.
    ApplicationElementSeq aeSeq[] =
    as.getTopLevelElements("AoAny");

    ...

    // Close the session.
    session.close();

  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_BASETYPE

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONSTRUCTURE_LISTELEMENTS

Purpose

Get the names of the application elements that match the pattern. The pattern is case sensitive and may contain wildcard characters.

Parameters

aePattern (Type = Pattern)

The name or the search pattern for the requested base elements.

Java Calling Sequence

Name[] applElemNames = applStruct.listElements(aePattern);

Returns:

Return-Name:	applElemNames
Return-Type:	NameSequence

The names of the application elements.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.ApplicationStructure;

  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");

  // Session successfully created ?
  if (session != null) {

     // Get application model.
           ApplicationStructure as =
            session.getApplicationStructure();

     // Get a list of application element names.
     String aeNameList[] = as.listElements("*");

     ...

     // Close the session.
     session.close();

  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONSTRUCTURE_LISTELEMENTSBYBASETYPE

Purpose

Get the names of application elements that are derived from the given base type.

Parameters

aeType (Type = BaseType)

The requested base type. The base element type can be a pattern.

Java Calling Sequence

Name[] applElemNames = applStruct.listElementsByBaseType(aeType);

Returns:

Return-Name:	applElemNames
Return-Type:	NameSequence

The names of the application elements.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.ApplicationStructure;

  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");

  // Session successfully created ?
  if (session != null) {

     // Get application model.
          ApplicationStructure as =
           session.getApplicationStructure();

     // Get a list of application element names.
     String aeNameList[] = as.listElementsByBaseType("AoTest");

     ...

     // Close the session.
     session.close();

  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_BASETYPE

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONSTRUCTURE_LISTENUMERATIONS

Purpose

Get the list of all enumeration names.

Parameters

None.

Java Calling Sequence

```
Name[] enumNames;
enumNames = applStruct.listEnumerations();
```

Returns:

Return-Name:	enumNames
Return-Type:	NameSequence

List with all enumeration names.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

APPLICATIONSTRUCTURE_LISTTOPLEVELELEMENTS

Purpose

Get the names of the top level application elements that are derived from the given base type. If the given base type is not a top level base element an exception is thrown. A top level application element is an application element without a father.

Parameters

aeType (Type = BaseType)

The requested base type.

Java Calling Sequence

Name[] applElemNames = applStruct.listTopLevelElements(aeType);

Returns:

Return-Name:	applElemNames
Return-Type:	NameSequence

The names of the application elements.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.ApplicationStructure;

  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");

  // Session successfully created ?
  if (session != null) {

     // Get application model.
          ApplicationStructure as =
           session.getApplicationStructure();

     // Get a list of top level application element names.
     String aeNameList[] = as.listTopLevelElements("*");

     ...

     // Close the session.
     session.close();

  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_BASETYPE

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONSTRUCTURE_REMOVEELEMENT

Purpose

Remove an application element from the application model.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

- Only allowed:
  - if the application element is empty (has no instances).
  - no relations with other application elements.

Parameters

applElem (Type = ApplicationElement)

The application element to be removed.

Java Calling Sequence

applStruct.removeElement(applElem);

Returns:

None.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.ApplicationStructure;
  import org.asam.ods.ApplicationElement;
  import org.asam.ods.BaseStructure;
  import org.asam.ods.BaseElement;
  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");
  // Session successfully created ?
  if (session != null) {
     // Get application model.
          ApplicationStructure as =
           session.getApplicationStructure();
     // Get base model.
          BaseStructure bs = session.getBaseStructure();
     // Get a base element by type.
     BaseElement be = bs.getElementByType("AoSubmatrix");
     // Create a new application element.
     ApplicationElement ae = as.createElement(be);
     ...
     // Remove an application element from application model.
     as.removeElement(ae);
     ...
     // Close the session.
```

```
        session.close();
    }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_HAS_INSTANCES

AO_HAS_REFERENCES

AO_IMPLEMENTATION_PROBLEM

AO_NOT_FOUND

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

APPLICATIONSTRUCTURE_REMOVEENUMERATIONDEFINITION

Purpose

Remove the enumeration definition. The server checks if the enumeration is still in use by one of the attributes. This method modifies the application model and is only allowed for the superuser.

Parameters

enumName (Type = T_STRING)

Name of the enumeration to remove.

Java Calling Sequence

```
applStruct.removeEnumerationDefinition(enumName);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

AO_ACCESS_DENIED

APPLICATIONSTRUCTURE_REMOVERELATION

Purpose

This method removes an application relation from the application model.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The elements of the relation are still part of the application model. If there are instances of the relation they are also removed.

Parameters

applRel (Type = ApplicationRelation)

The application relation to be removed.

Java Calling Sequence

applStruct.removeRelation(applRel);

Returns:

None.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.ApplicationStructure;
  import org.asam.ods.ApplicationRelation;
  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");
  // Session successfully created ?
  if (session != null) {
     // Get application model.
           ApplicationStructure as =
           session.getApplicationStructure();
     // Create a new application relation.
     ApplicationRelation ar = as.createRelation();
     ...
     // Remove an application relation from an application
        model.
     as.removeRelation(ar);
     ...
     // Close the session.
     session.close();
  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_HAS_INSTANCES

AO_IMPLEMENTATION_PROBLEM

AO_NOT_FOUND

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.8 BaseAttribute

BaseAttribute_getBaseElement

Purpose

Return the base element to which the attibute belongs..

Parameters

None.

Java Calling Sequence

```
BaseElement baseElem = baseAttr.getBaseElement();
```

Returns:

Return-Name:	baseElem
Return-Type:	BaseElement

The base element of the attribute.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASEATTRIBUTE_GETDATATYPE

Purpose

Get the data type of the base attribute.

Parameters

None.

Java Calling Sequence

DataType baDataType = baseAttr.getDataType();

Returns:

Return-Name:	baDataType
Return-Type:	DataType

The data type of the base attribute.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASEATTRIBUTE_GETNAME

Purpose

Get the name of the base attribute.

Parameters

None.

Java Calling Sequence

```
Name baName = baseAttr.getName();
```

Returns:

Return-Name:	baName
Return-Type:	Name

The name of the base attribute.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASEATTRIBUTE_ISOBLIGATORY

Purpose

Get the obligatory flag of the base attribute.

Parameters

None.

Java Calling Sequence

```
T_BOOLEAN baIsObligatory = baseAttr.isObligatory();
```

Returns:

Return-Name:	baIsObligatory
Return-Type:	T_BOOLEAN

The obligatory flag of the base attribute.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASEATTRIBUTE_ISUNIQUE

Purpose

Get the unique flag of the base attribute.

Parameters

None.

Java Calling Sequence

```
T_BOOLEAN baIsUnique = baseAttr.isUnique();
```

Returns:

Return-Name:	baIsUnique
Return-Type:	T_BOOLEAN

The unique flag of the base attribute.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.9 BASEELEMENT

BASEELEMENT_GETALLRELATIONS

Purpose

Get all known relations of the base element.

Parameters

None.

Java Calling Sequence

BaseRelation[] baseRels = baseElem.getAllRelations();

Returns:

Return-Name:	baseRels
Return-Type:	BaseRelationSequence

All known relations of the base element.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BaseElement_getAttributes

Purpose

Get attributes of the base element.

Parameters

baPattern (Type = Pattern)

The name or the search pattern for the requested base attributes.

Java Calling Sequence

```
BaseAttribute[] baseAttrs = baseElem.getAttributes(baPattern);
```

Returns:

Return-Name:	baseAttrs
Return-Type:	BaseAttributeSequence

The requested attributes of the base element.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASEELEMENT_GETRELATEDELEMENTSBYRELATIONSHIP

Purpose

Get the related elements of a base element defined by the relationship.

Parameters

brRelationship (Type = Relationship)

The requested relationship.

Java Calling Sequence

```
BaseElement[] baseElems =
baseElem.getRelatedElementsByRelationship(brRelationship);
```

Returns:

Return-Name:	baseElems
Return-Type:	BaseElementSequence

The related elements of a base element.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_RELATIONSHIP

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BaseElement_getRelationsByType

Purpose

Get the base element's relations of the requested relation type.

Parameters

brRelationType (Type = RelationType)

The requested relation type.

Java Calling Sequence

```
BaseRelation[] baseRels = baseElem.getRelationsByType(brRelationType);
```

Returns:

Return-Name:	baseRels
Return-Type:	BaseRelationSequence

The base element's relations of the requested type.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_RELATION_TYPE

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASEELEMENT_GETTYPE

Purpose

Get the type of the base element. The type of the base element is identical with the name of the base element. The type of the base element is a string.

Parameters

None.

Java Calling Sequence

```
BaseType beType = baseElem.getType();
```

Returns:

Return-Name:	beType
Return-Type:	BaseType

The type of the base element.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASEELEMENT_ISTOPLEVEL

Purpose

Get whether or not the base element is a top level element. Top level elements are elements without a father.

Parameters

None.

Java Calling Sequence

```
T_BOOLEAN beIsTopLevel = baseElem.isTopLevel();
```

Returns:

Return-Name:	beIsTopLevel
Return-Type:	T_BOOLEAN

Boolean whether or not the base element is a top level element.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASEELEMENT_LISTATTRIBUTES

Purpose

Get attribute names of the base element.

Parameters

baPattern (Type = Pattern)

The name or the search pattern for the requested base attribute names.

Java Calling Sequence

```
Name[] baseElemNames = baseElem.listAttributes(baPattern);
```

Returns:

Return-Name:	baseElemNames
Return-Type:	NameSequence

The requested attribute names of the base element.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BaseElement_listRelatedElementsByRelationship

Purpose

Get the related element names of the base element defined by the relationship.

Parameters

brRelationship (Type = Relationship)

The requested relationship.

Java Calling Sequence

```
BaseType[] baseTypes = baseElem.listRelatedElementsByRelationship(brRelationship);
```

Returns:

Return-Name:	baseTypes
Return-Type:	BaseTypeSequence

The related element names of the base element.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_RELATIONSHIP

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.10 BASERELATION

BASERELATION_GETELEM1

Purpose

Get the first base element of the base relation.

Parameters

None.

Java Calling Sequence

```
BaseElement baseElem = baseRel.getElem1();
```

Returns:

Return-Name:	baseElem
Return-Type:	BaseElement

The first base element of the base relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASERELATION_GETELEM2

Purpose

Get the second base element of the base relation.

Parameters

None.

Java Calling Sequence

```
BaseElement baseElem = baseRel.getElem2();
```

Returns:

Return-Name:	baseElem
Return-Type:	BaseElement

The second base element of the base relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASERELATION_GETINVERSERELATIONRANGE

Purpose

Get the inverse relation range of the base relation.

Parameters

None.

Java Calling Sequence

```
RelationRange brRange = baseRel.getInverseRelationRange();
```

Returns:

Return-Name:	brRange
Return-Type:	RelationRange

The inverse relation range of the base relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BaseRelation_getInverseRelationship

Purpose

Get the inverse relationship of the base relation.

Parameters

None.

Java Calling Sequence

```
Relationship relationship = baseRel.getInverseRelationship();
```

Returns:

Return-Name:	relationship
Return-Type:	Relationship

The inverse relationship of the base relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASERELATION_GETRELATIONNAME

Purpose

Get the relation name of the base relation.

Parameters

None.

Java Calling Sequence

```
Name brName = baseRel.getRelationName();
```

Returns:

Return-Name:	brName
Return-Type:	Name

The relation name of the base relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BaseRelation_getRelationRange

Purpose

Get the relation range of the base relation.

Parameters

None.

Java Calling Sequence

```
RelationRange brRange = baseRel.getRelationRange();
```

Returns:

Return-Name:	brRange
Return-Type:	RelationRange

The relation range of the base relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASERELATION_GETRELATIONSHIP

Purpose

Get the relationship of the base relation.

Parameters

None.

Java Calling Sequence

Relationship relationship = baseRel.getRelationship();

Returns:

Return-Name:	relationship
Return-Type:	Relationship

The relationhip of the base relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BaseRelation_getRelationType

Purpose

Get the relation type of the base relation.

Parameters

None.

Java Calling Sequence

```
RelationType brType = baseRel.getRelationType();
```

Returns:

Return-Name:	brType
Return-Type:	RelationType

The relation type of the base relation.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.11 BASESTRUCTURE

BaseStructure is the interface that allows to access the ASAM ODS base model which currently is used by the server (please note the difference in the wording between structure and model). This is similar to ApplicationStructure versus ASAM ODS application model.

BASESTRUCTURE_GETELEMENTBYTYPE

Purpose

Get the base element that matches the requested type. The type of a base element is identical with the name of the base element.

Parameters

beType (Type = BaseType)

The name of the requested base element.

Java Calling Sequence

BaseElement baseElem = baseStruct.getElementByType(beType);

Returns:

Return-Name:	baseElem
Return-Type:	BaseElement

The requested base element.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.BaseStructure;

  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");

  // Session successfully created ?
  if (session != null) {

     // Get base model.
          BaseStructure bs = session.getBaseStructure();

     // Get a the base element of type AoMeasurement.
     BaseElement be = bs.getElementByType("AoMeasurement");
     ...
     // Close the session.
     session.close();

  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_BASETYPE

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASESTRUCTURE_GETELEMENTS

Purpose

Get the base elements that match the pattern. The pattern is case sensitive and may contain wildcard characters.

Parameters

bePattern (Type = Pattern)

The name or the search pattern for the requested base elements.

Java Calling Sequence

BaseElement[] baseElems = baseStruct.getElements(bePattern);

Returns:

Return-Name:	baseElems
Return-Type:	BaseElementSequence

The requested base elements.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.BaseStructure;

  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");

  // Session successfully created ?
  if (session != null) {

     // Get base model.
          BaseStructure bs = session.getBaseStructure();

     // Get a list of base elements.
     BaseElement beList[] = bs.getElements("*");

     ...

     // Close the session.
     session.close();

  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASESTRUCTURE_GETRELATION

Purpose

Get the base relation between two base elements.

Parameters

elem1 (Type = BaseElement)

The base element from which the relation starts.

elem2 (Type = BaseElement)

The base element to which the relation points.

Java Calling Sequence

BaseRelation baseRel = baseStruct.getRelation(elem1,elem2);

Returns:

Return-Name:	baseRel
Return-Type:	BaseRelation

The base relation between the two base elements.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.BaseStructure;

  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");
  // Session successfully created ?
  if (session != null) {
     // Get base model.
     BaseStructure bs = session.getBaseStructure();
     // Get a the base elements of type AoMeasurement and
        AoQuantity.
     BaseElement be1 = bs.getElementByType("AoMeasurement");
     BaseElement be2 = bs.getElementByType("AoQuantity");
     // Get the relation between the elements.
     BaseRelation br = bs.getRelation(be1,be2);

     ...

     // Close the session.
     session.close();

  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASESTRUCTURE_GETTOPLEVELELEMENTS

Purpose

Get the top level base elements that match the pattern. The pattern is case sensitive and may contain wildcard characters. A top level base element is a base element without a father.

Parameters

bePattern (Type = Pattern)

The name or the search pattern for the requested top level base elements.

Java Calling Sequence

BaseElement[] baseElems = baseStruct.getTopLevelElements(bePattern);

Returns:

Return-Name: baseElems

Return-Type: BaseElementSequence

The requested top level base elements.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.BaseStructure;

  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");

  // Session successfully created ?
  if (session != null) {

     // Get base model.
          BaseStructure bs = session.getBaseStructure();

     // Get a list of top level base elements.
     BaseElement beList[] = bs.getTopLevelElements("*");

     ...

     // Close the session.
     session.close();

  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASESTRUCTURE_GETVERSION

Purpose

Get the version of the base model.The version of the base model is the version of the ASAM ODS base model.

Parameters

None.

Java Calling Sequence

T_STRING version = baseStruct.getVersion();

Returns:

Return-Name: version

Return-Type: T_STRING

The version of the ASAM ODS base model.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.BaseStructure;

  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");

  // Session successfully created ?
  if (session != null) {

     // Get base model.
           BaseStructure bs = session.getBaseStructure();

     // Get current base model version.
     String bsVersion = bs.getVersion();

     ...

     // Close the session.
     session.close();

  }
```

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASESTRUCTURE_LISTELEMENTS

Purpose

Get the base element names that match the pattern. The pattern is case sensitive and may contain wildcard characters.

Parameters

bePattern (Type = Pattern)

The name or the search pattern for the requested base element names.

Java Calling Sequence

BaseType[] baseTypes = baseStruct.listElements(bePattern);

Returns:

Return-Name:	baseTypes
Return-Type:	BaseTypeSequence

The requested base element names.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.BaseStructure;

  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");

  // Session successfully created ?
  if (session != null) {

     // Get base model.
          BaseStructure bs = session.getBaseStructure();

     // Get a list of base element names.
     String beNameList[] = bs.ListElements("*");

     ...

     // Close the session.
     session.close();

  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BASESTRUCTURE_LISTTOPLEVELELEMENTS

Purpose

Get the top level base element names that match the pattern. The pattern is case sensitive and may contain wildcard characters. A top level base element is a base element without a father.

Parameters

bePattern (Type = Pattern)

The name or the search pattern for the requested top level base element names.

Java Calling Sequence

BaseType[] baseTypes = baseStruct.listTopLevelElements(bePattern);

Returns:

Return-Name: baseTypes

Return-Type: BaseTypeSequence

The requested top level base element names.

Examples:

Language: Java

```
import org.asam.ods.AoSession;
  import org.asam.ods.BaseStructure;

  // Create a new session with aoFactory.
  AoSession session;
  session = aoFactory.newSession("");

  // Session successfully created ?
  if (session != null) {

     // Get base model.
          BaseStructure bs = session.getBaseStructure();

     // Get a list of top level base element names.
     String beNameList[] = bs.ListTopLevelElements("*");

     ...

     // Close the session.
     session.close();

  }
```

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.12 BLOB

BLOB_APPEND

Purpose

Append a byte sequence to the binary large object.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

Parameters

value (Type = S_BYTE)

The byte sequence.

Java Calling Sequence

```
blob.append(value);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BLOB_COMPARE

Purpose

Compares the content of the binary large object. The headers are not compared.

Parameters

aBlob (Type = T_BLOB)

The blob to compare.

Java Calling Sequence

T_BOOLEAN contentEqual = blob.compare(aBlob);

Returns:

Return-Name:	contentEqual
Return-Type:	T_BOOLEAN

A flag whether or not the compared blobs are equal.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BLOB_DESTROY

Purpose

Destroy the object on the server. This method is used to tell the server that the blob object is not used anymore by the client. Access to this object after the destroy method will lead to an exception.

Parameters

None.

Java Calling Sequence

```
blob.destroy()
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

BLOB_GET

Purpose

Get a part of the binary large object.

Parameters

offset (Type = T_LONG)

The starting position of the data in the blob.

length (Type = T_LONG)

The number of bytes requested from the blob.

Java Calling Sequence

```
S_BYTE byteStream = blob.get(offset,length);
```

Returns:

Return-Name:	byteStream
Return-Type:	S_BYTE

The request part of the blob data.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BLOB_GETHEADER

Purpose

Get the header of the binary large object.

Parameters

None.

Java Calling Sequence

```
T_STRING header = blob.getHeader();
```

Returns:

Return-Name:	header
Return-Type:	T_STRING

The blob header.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BLOB_GETLENGTH

Purpose

Get the length of the binary large object without loading it.

Parameters

None.

Java Calling Sequence

T_LONG length = blob.getLength();

Returns:

Return-Name: length

Return-Type: T_LONG

The blob length.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BLOB_SET

Purpose

Clear the binary large object and set the new data.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

Parameters

value (Type = S_BYTE)

The new blob data.

Java Calling Sequence

blob.set(value);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

BLOB_SETHEADER

Purpose

Set the header of a binary large object.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

Parameters

header (Type = T_STRING)

The blob header.

Java Calling Sequence

```
blob.setHeader(header);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.13 COLUMN

COLUMN_DESTROY

Purpose

Destroy the object on the server. This method is used to tell the server that this object is not used anymore by the client. Access to this object after the destroy method will lead to an exception.

Parameters

None.

Java Calling Sequence

column.destroy()

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

COLUMN_GETDATATYPE

Purpose

Get the data type of the column.

Parameters

None.

Java Calling Sequence

```
DataType dataType = column.getDataType();
```

Returns:

Return-Name:	dataType
Return-Type:	DataType

The data type of the column.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

COLUMN_GETFORMULA

Purpose

Get the formula of the column.

Parameters

None.

Java Calling Sequence

```
T_STRING formula = column.getFormula();
```

Returns:

Return-Name:	formula
Return-Type:	T_STRING

The formula of the column.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

COLUMN_GETNAME

Purpose

Get the name of the column.

Parameters

None.

Java Calling Sequence

```
Name columnName = column.getName();
```

Returns:

Return-Name:	columnName
Return-Type:	Name

The name of the column.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

COLUMN_GETSOURCEMQ

Purpose

Get the source measurement quantity.

Parameters

None.

Java Calling Sequence

```
InstanceElement instElem = column.getSourceMQ();
```

Returns:

Return-Name:	instElem
Return-Type:	InstanceElement

The source measurement quantity.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

COLUMN_GETUNIT

Purpose

Get the unit of the column.

Parameters

None.

Java Calling Sequence

```
T_STRING unit = column.getUnit();
```

Returns:

Return-Name:	unit
Return-Type:	T_STRING

The unit of the column.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

COLUMN_ISINDEPENDENT

Purpose

Is the column an independent column

Parameters

None.

Java Calling Sequence

```
T_BOOLEAN independent = column.isIndependent();
```

Returns:

Return-Name:	independent
Return-Type:	T_BOOLEAN

The independent flag of the column.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

COLUMN_ISSCALING

Purpose

Is the column an scaling column

Parameters

None.

Java Calling Sequence

```
T_BOOLEAN scaling = column.isScaling();
```

Returns:

Return-Name:	scaling
Return-Type:	T_BOOLEAN

Tells if the column is a scaling column.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

COLUMN_SETFORMULA

Purpose

Set the formula of the column.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

Parameters

formula (Type = T_STRING)

The formula.

Java Calling Sequence

column.setFormula(formula);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

COLUMN_SETINDEPENDENT

Purpose

Set the column as an indepent column.

Parameters

independent (Type = T_BOOLEAN)

The new value of the independent flag.

Java Calling Sequence

column.setIndependent(independent);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

COLUMN_SETSCALING

Purpose

Set the column to a scaling column.

Parameters

scaling (Type = T_BOOLEAN)

The new value of the scaling flag.

Java Calling Sequence

column.setScaling(scaling);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

COLUMN_SETUNIT

Purpose

Set the unit of the column. This is only a temporary conversion unit when getting the data of a column. This unit is not stored in the database. If a permanent storage of the conversion unit is required the corresponding measurement quantity needs to be changed.

Parameters

unit (Type = T_STRING)

The physical unit.

Java Calling Sequence

```
column.setUnit(unit);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.14 ELEMRESULTSETEXTSEQITERATOR

ELEMRESULTSETEXTSEQITERATOR_DESTROY

Purpose

Destroy the iterator and free the associated memory.

Parameters

None.

Java Calling Sequence

```
ersIter.destroy();
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

ElemResultSetExtSeqIterator_getCount

Purpose

Get the total number of elements accessible by the iterator.

Parameters

None.

Java Calling Sequence

T_LONG cnt = ersIter.getCount();

Returns:

Return-Name:	count
Return-Type:	T_LONG

The number of elements accessible by the iterator.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

ELEMRESULTSETEXTSEQITERATOR_NEXTN

Purpose

Get the next n elements from the sequence.

Parameters

how_many (Type = T_LONG)

The number of requested elements.

Java Calling Sequence

```
T_LONG how_many = 10;
ElemResultSetExt[] elemResults = ersIter.nextN(how_many)
```

Returns:

Return-Name:	elemResults
Return-Type:	ElemResultSetExtSequence

The next n attribute values from the element result set sequence.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

ElemResultSetExtSeqIterator_nextOne

Purpose

Get the next element from the sequence.

Parameters

None.

Java Calling Sequence

```
ElemResultSetExt elemResult = ersIter.nextOne()
```

Returns:

Return-Name:	elemResult
Return-Type:	ElemResultSetExt

The next attribute values from the element result set sequence.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

ELEMRESULTSETEXTSEQITERATOR_RESET

Purpose

Reset the pointer in the element sequence to the first element.

Parameters

None.

Java Calling Sequence

```
ersIter.reset();
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.15 ENUMERATIONDEFINITION

ENUMERATIONDEFINITION_ADDITEM

Purpose

Add a new item to the enumeration. This method modifies the application model and is only allowed for the superuser.

The name of an item must not exceed the maximum name length of the underlying physical storage. The current specification of the physical storage restricts the length of enumeration items to a maximum of 128 characters.

Parameters

itemName (Type = T_STRING)

The name of the new item.

Java Calling Sequence

```
enumDef.addItem(itemName);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

AO_ACCESS_DENIED

AO_DUPLICATE_NAME

AO_DUPLICATE_VALUE

ENUMERATIONDEFINITION_GETINDEX

Purpose

Get the index of the enumeration.

Parameters

None.

Java Calling Sequence

```
T_LONG enumIndex = enumDef.getIndex();
```

Returns:

Return-Name:	enumIndex
Return-Type:	T_LONG

The index of the enumeration.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
AO_TRANSACTION_NOT_ACTIVE

ENUMERATIONDEFINITION_GETITEM

Purpose

Get the value of an item.

Parameters

itemName (Type = T_STRING)

The name of the item.

Java Calling Sequence

```
T_LONG item = enumDef.getItem(itemName);
```

Returns:

Return-Name:	item
Return-Type:	T_LONG

The number of the item.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

AO_NOT_FOUND

ENUMERATIONDEFINITION_GETITEMNAME

Purpose

Get the name of an item.

Parameters

item (Type = T_LONG)

The value of the item.

Java Calling Sequence

```
T_STRING itemName = enumDef.getItemName(item);
```

Returns:

Return-Name:	itemName
Return-Type:	T_STRING

The name of the item.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
AO_TRANSACTION_NOT_ACTIVE
AO_NOT_FOUND

ENUMERATIONDEFINITION_GETNAME

Purpose

Get the name of the enumeration.

Parameters

None.

Java Calling Sequence

```
T_STRING enumName = enumDef.getName();
```

Returns:

Return-Name:	enumName
Return-Type:	T_STRING

Name of the enumeration.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

ENUMERATIONDEFINITION_LISTITEMNAMES

Purpose

List the possible names of the enumeration. The sort order of the list is the value of the item. The first item has a value of 0 (zero).

Parameters

None.

Java Calling Sequence

```
Name[] nameList = enumDef.listItemNames();
```

Returns:

Return-Name:	nameList
Return-Type:	NameSequence

List with all possiable names of the enumeration items.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

ENUMERATIONDEFINITION_RENAMEITEM

Purpose

Rename the item of the enumeration. This method modifies the application model and is only allowed for the superuser.

Parameters

oldItemName (Type = T_STRING)

The existing name of the itrem.

newItemName (Type = T_STRING)

the new name of the item.

Java Calling Sequence

```
enumDef.renameItem(oldItemName, newItemName);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

AO_NOT_FOUND

ENUMERATIONDEFINITION_SETNAME

Purpose

Set the name of the enumeration. This method modifies the application model and is only allowed for the superuser.

The name of an enumeration definition must not exceed the maximum name length of the underlying physical storage.The current specification of the physical storage restricts the length of enumeration names to a maximum of 30 characters.

Parameters

enumName (Type = T_STRING)

Name of the enumeration.

Java Calling Sequence

```
enumDef.setName(enumName);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

AO_ACCESS_DENIED

10.2.16 INSTANCEELEMENT

INSTANCEELEMENT_ADDINSTANCEATTRIBUTE

Purpose

Add an instance attribute to the instance. The instance attribute is built as a Name/Value/Unit tuple on the client. This method has to copy the data from the client to the server. The name of the attribute must be unique.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

Parameters

instAttr (Type = NameValueUnit)

The instance attribute to be added.

Java Calling Sequence

```
instElem.addInstanceAttribute(instAttr);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_COMPARE

Purpose

Compare an instance element. The Ids of the application elements and the Ids of the instance elements are compared. The Ids of the application elements will be compare first.

Parameters

compIeObj (Type = InstanceElement)

The instance element to compare with.

Java Calling Sequence

T_LONGLONG diff = ieObj.compare(compIeObj);

Returns:

Return-Name:	diff
Return-Type:	T_LONGLONG

The difference of the Id's. Meaning:

diff < 0 ElemId of instance is smaller then instance to compare with.

diff == 0 ElemId is identical.

diff > 0 ElemId of instance is greater then instance to compare with.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_CREATERELATEDINSTANCES

Purpose

Create a list with instances which are related to the actual instance element. The attribute are given with the name of the sequence. The values of the attributes are given in the value sequence. The index in the different value sequences match for one instance element. The index in the instance element sequence of the related instances match for the instance with the same index in the value sequence.

Parameters

applRel (Type = ApplicationRelation)

The application relation for wich the related instances will be created.^

attributes (Type = NameValueSeqUnitSequence)

The attributes of the new created instances.

relatedInstances (Type = ApplicationRelationInstanceElementSeqSequence)

The list with related instances for different application relations.

Java Calling Sequence

```
InstanceElement[] instElems = ieObj.createRelatedInstances(applRel, attributes,
                              relatedInstances);
```

Returns:

Return-Name:	instElems
Return-Type:	InstanceElementSequence

The list with the new created instances.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

AO_INVALID_REQUEST

INSTANCEELEMENT_CREATERELATION

Purpose

Create a relation between the current and the given instance. Check if the application elements of the relation matches the application elements of the instances.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

Parameters

relation (Type = ApplicationRelation)

The application relation.

instElem (Type = InstanceElement)

The instance element.

Java Calling Sequence

```
instElem.createRelation(relation,instElem);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_RELATION

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_DEEPCOPY

Purpose

Provides an easy-to-use and effective copy mechanismn for instance element hierarchies inside the server (e.g. copy a project with all tests or copy a test with all measurements). The deep copy follows only the child references but not the informational references. Example: Copying elements of type AoMeasurement does not include copying the referenced elements of type AoMeasurementQuantity. The copied instance elements of type AoMeasurement will reference the same measurement quantities as the original. An application that wants to copy the measurement quantity also must do this (including setting the proper references) by itself e.g. with another call to shallowCopy; deepCopy is not necessary in this case because AoMeasurementQuantity has no children.

Parameters

newName (Type = T_STRING)

The name of the new instance element. If a new version shall be created this parameter may be NULL to use the same name for the copy. In this case a new version must be provided.

newVersion (Type = T_STRING)

The version of the new instance element. This parameter may be NULL if a new name is provided.

Java Calling Sequence

InstanceElement newInstElem = instElem.deepCopy(newName,newVersion);

Returns:

Return-Name:	newInstElem
Return-Type:	InstanceElement

The reference to the copied instance element hierarchy.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

INSTANCEELEMENT_DESTROY

Purpose

Destroy the object on the server. This method is used to tell the server that this object is not used anymore by the client. Access to this object after the destroy method will lead to an exception.

Parameters

None.

Java Calling Sequence

ieObj.destroy()

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

INSTANCEELEMENT_GETAPPLICATIONELEMENT

Purpose

Get the application element of the instance element.

Parameters

None.

Java Calling Sequence

ApplicationElement applElem = instElem.getApplicationElement();

Returns:

Return-Name:	applElem
Return-Type:	ApplicationElement

The application element from which the instance element is derived.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_GETASAMPATH

Purpose

Get the ASAM-Path of the instance element.

Parameters

None.

Java Calling Sequence

```
Name asamPath = instElem.getAsamPath();
```

Returns:

Return-Name:	asamPath
Return-Type:	Name

The ASAM path to the instance element.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_GETID

Purpose

Get the Id of the instance element.

Parameters

None.

Java Calling Sequence

T_LONGLONG ieId = instElem.getId();

Returns:

Return-Name:	ieId
Return-Type:	T_LONGLONG

The Id of the instance element.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_GETINITIALRIGHTS

Purpose

Retrieve access control list information for the initial rights of the given object.

Parameters

None.

Java Calling Sequence

```
InitialRight[] initialRights = instElem.getInitialRights();
```

Returns:

Return-Name:	initialRights
Return-Type:	InitialRightSequence

The access control list entries with the initial rights of the given application element.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

INSTANCEELEMENT_GETNAME

Purpose

Get the name of the instance element.

Parameters

None.

Java Calling Sequence

```
Name ieName = instElem.getName();
```

Returns:

Return-Name:	ieName
Return-Type:	Name

The name of the instance element.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_GETRELATEDINSTANCES

Purpose

Get the related instances. The application relation and the name of the related instances specify the listed instances. The pattern is case sensitive and may contain wildcard characters.

Parameters

applRel (Type = ApplicationRelation)

The application relation.

iePattern (Type = Pattern)

The name or the search pattern for the related instance names.

Java Calling Sequence

```
InstanceElementIterator ieIterator = instElem.getRelatedInstances(applRel,iePattern);
```

Returns:

Return-Name:	ieIterator
Return-Type:	InstanceElementIterator

The related instances.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_RELATION

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_GETRELATEDINSTANCESBYRELATIONSHIP

Purpose

Get the list of related instances. The relationship and the name of the related instances specify the listed instances. The pattern is case sensitive and may contain wildcard characters.

Parameters

ieRelationship (Type = Relationship)

The requested relationship.

iePattern (Type = Pattern)

The name or the search pattern for the related instance names.

Java Calling Sequence

```
InstanceElementIterator ieIterator =
instElem.getRelatedInstancesByRelationship(ieRelationship,iePattern);
```

Returns:

Return-Name:	ieIterator
Return-Type:	InstanceElementIterator

The related instances.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_RELATIONSHIP

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_GETRIGHTS

Purpose

Retrieve access control list information of the given object.

Parameters

None.

Java Calling Sequence

```
ACL[] aclEntries = instElem.getRights();
```

Returns:

Return-Name:	aclEntries
Return-Type:	ACLSequence

The access control list entries of the given application element.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

INSTANCEELEMENT_GETVALUE

Purpose

Get the attribute value (name, value and unit) of the given attribute of the instance element. This method will not return the value of relation attributes, use the method getRelatedInstances.

Parameters

attrName (Type = Name)

The name of the requested attribute.

Java Calling Sequence

NameValueUnit nameValueUnit = instElem.getValue(attrName);

Returns:

Return-Name:	nameValueUnit
Return-Type:	NameValueUnit

The attribute value.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_GETVALUEBYBASENAME

Purpose

Get the attribute value (value and unit) of the attribute inherited from the given base attribute of the instance element. The base name is case insensitive and may not contain wildcard characters.

Parameters

baseAttrName (Type = Name)

The base name of the requested attribute.

Java Calling Sequence

NameValueUnit nameValueUnit = instElem.getValueByBaseName(baseAttrName);

Returns:

Return-Name:	nameValueUnit
Return-Type:	NameValueUnit

The attribute value.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_GETVALUEINUNIT

Purpose

Get the attribute value (name, value and unit) of the given attribute of the instance element.

Parameters

attr (Type = NameUnit)

The name of the requested attribute and the unit of the attribute value.

Java Calling Sequence

```
NameValueUnit nameValueUnit = instElem.getValueInUnit(attr);
```

Returns:

Return-Name:	nameValueUnit
Return-Type:	NameValueUnit

The attribute value, value converted to the requested unit.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_INCOMPATIBLE_UNITS

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_GETVALUESEQ

Purpose

Get the sequence of the values of an application or instance attribute, specified by their names. The name sequence can use a pattern (*) for all attributes of the instance element. This means that application as well as instance attributes will be delivered.

Parameters

attrNames (Type = NameSequence)

The names of the attributes to be reported.

Java Calling Sequence

```
Name attrNames[];
...
NameValueUnit values[] = ieObj.getValueSeq(attrNames);
```

Returns:

Return-Name:	values
Return-Type:	NameValueUnitSequence

The sequence of the attribute values.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_LISTATTRIBUTES

Purpose

Get the attribute names from the instance element. The attributes reserved for a relation are not listed.

Parameters

iaPattern (Type = Pattern)

The name or the search pattern for the attribute names.

aType (Type = AttrType)

The requested attribute type.

Java Calling Sequence

```
Name[] ieNames = instElem.listAttributes(iaPattern,aType);
```

Returns:

Return-Name:	ieNames
Return-Type:	NameSequence

The names of the attributes.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_ATTRIBUTE_TYPE

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_LISTRELATEDINSTANCES

Purpose

Get the names of the related instances. The application relation and the name of the related instances specifies the listed names. The pattern is case sensitive and may contain wildcard characters.

Parameters

ieRelation (Type = ApplicationRelation)

The application relation.

iePattern (Type = Pattern)

The name or the search pattern for the related instance names.

Java Calling Sequence

```
NameIterator ieNameIterator = instElem.listRelatedInstances(ieRelation,iePattern);
```

Returns:

Return-Name:	ieNameIterator
Return-Type:	NameIterator

The names of the related instances.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_LISTRELATEDINSTANCESBYRELATIONSHIP

Purpose

Get the names of the related instances. The relationship and the name of the related instances specify the listed names. The pattern is case sensitive and may contain wildcard characters.

Parameters

ieRelationship (Type = Relationship)

The requested relationship.

iePattern (Type = Pattern)

The name or the search pattern for the related instance names.

Java Calling Sequence

```
NameIterator ieNameIterator =
instElem.listRelatedInstancesByRelationship(ieRelationship,iePattern);
```

Returns:

Return-Name:	ieNameIterator
Return-Type:	NameIterator

The names of the related instances.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_RELATIONSHIP

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_REMOVEINSTANCEATTRIBUTE

Purpose

Remove an instance attribute.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The application attributes can't be removed.

Parameters

attrName (Type = Name)

The name of the attribute to be removed.

Java Calling Sequence

```
instElem.removeInstanceAttribute(attrName);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_FOUND

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_REMOVERELATION

Purpose

Remove the relation between the current instance and the given instance. It is necessary to specify the instance element in case of n:m relations if not all relations shall be deleted.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

Parameters

applRel (Type = ApplicationRelation)

The relation to be removed.

instElem_nm (Type = InstanceElement)

The instance element for specific delete from n:m relations.

Java Calling Sequence

```
instElem.removeRelation(applRel,instElem_nm);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_RELATION

AO_NOT_FOUND

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_RENAMEINSTANCEATTRIBUTE

Purpose

Rename the instance attribute. The application attributes can't be renamed.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

Parameters

oldName (Type = Name)

The old instance attribute name.

newName (Type = Name)

The new instance attribute name.

Java Calling Sequence

```
instElem.renameInstanceAttribute(oldName,newName);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_DUPLICATE_NAME

AO_IMPLEMENTATION_PROBLEM

AO_NOT_FOUND

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_SETINITIALRIGHTS

Purpose

The given usergroup the initial rights should be set for. <rights> defines the rights to set or to clear. If the parameter <set> is set to 'set', the rights in <rights> are set, all others are cleared. If the parameter <set> is set to 'add', the rights in <rights> are added to the existing rights. If the parameter <set> is set to 'remove', the rights in <rights> are removed from the existing rights.

Parameters

usergroup (Type = InstanceElement)

The usergroup for which the initial rights will be modified.

rights (Type = T_LONG)

The new initial rights for the usergroup. The rights constants are defined in the interface SecurityRights. The interface definition language IDL does not allow to set the values of enumerations; thus the constant definitions have to be done in the interface.

refAid (Type = T_LONGLONG)

The Id of referencing application element for which the initial rights will be used. If no refAid is set the initial rights will be used for each new instance element independent of the application element.

set (Type = RightsSet)

What to do with the new initial rights.

Java Calling Sequence

```
instElem.setInitialRights(usergroup,rights, refAid,set);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

INSTANCEELEMENT_SETNAME

Purpose

Set the name of an instance element.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

Parameters

iaName (Type = Name)

The instance attribute name.

Java Calling Sequence

```
instElem.setName(iaName);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_SETRIGHTS

Purpose

The given usergroup the rights should be set for. <rights> defines the rights to set or to clear. If the parameter <set> is set to 'set', the rights in <rights> are set, all others are cleared. If the parameter <set> is set to 'add', the rights in <rights> are added to the existing rights. If the parameter <set> is set to 'remove', the rights in <rights> are removed from the existing rights.

Parameters

usergroup (Type = InstanceElement)

The usergroup for which the rights will be modified.

rights (Type = T_LONG)

The new right for the usergroup. The rights constants are defined in the interface SecurityRights. The interface definition language IDL does not allow to set the values of enumerations thus the constant definitions had to be done in an interface.

set (Type = RightsSet)

What to do with the new right.

Java Calling Sequence

```
instElem.setRights(usergroup,rights,set);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

INSTANCEELEMENT_SETVALUE

Purpose

Set the value of an application or instance attribute.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The name of the attribute is specified by the name of the NameValueUnit tuple. If the application attribute flag unique is set, the uniqueness of the new value is checked.

This method can not be used to set the valueof a relation attribute, use the method createRelation.

Parameters

value (Type = NameValueUnit)

The value to be set in the instance element.

Java Calling Sequence

```
instElem.setValue(value);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_DUPLICATE_VALUE

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_SETVALUESEQ

Purpose

Set a sequences of values of an application or instance attributes.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The name of the attribute is specified by the name of the NameValueUnit tuple. If the application attribute flag unique is set, the uniqueness of the new value is checked.

Parameters

values (Type = NameValueUnitSequence)

The sequence of the values to be set at the instance element.

Java Calling Sequence

instElem.setValueSeq(values);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_DUPLICATE_VALUE

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENT_SHALLOWCOPY

Purpose

Provides an easy-to-use and effective copy mechanismn for instance elements inside the server. The new instance elements gets a copy of all attribute values and informational relations that are available in the original instance element. The new instance element has the same parent as the original instance element but it does not have references to any children of the original instance element.

Parameters

newName (Type = T_STRING)

The name of the new instance element. If a new version shall be created this parameter may be NULL to use the same name for the copy. In this case a new version must be provided.

newVersion (Type = T_STRING)

The version of the new instance element. This parameter may be NULL if a new name is provided.

Java Calling Sequence

```
InstanceElement newInstElem = instElem.shallowCopy(newName,newVersion);
```

Returns:

Return-Name:	newInstElem
Return-Type:	InstanceElement

The reference to the copied instance element.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

INSTANCEELEMENT_UPCASTMEASUREMENT

Purpose

Cast an instance element to a measurement. There are some object-oriented languages which do not allow this cast.

Parameters

None.

Java Calling Sequence

```
Measurement measurement = instElem.upcastMeasurement();
```

Returns:

Return-Name:	measurement
Return-Type:	Measurement

The instance of type measurement.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_INVALID_BASETYPE

INSTANCEELEMENT_UPCASTSUBMATRIX

Purpose

Cast an instance element to a submatrix. There are some object-oriented languages which do not allow this cast.

Parameters

None.

Java Calling Sequence

SubMatrix subMatrix = instElem.upcastSubMatrix();

Returns:

Return-Name: subMatrix

Return-Type: SubMatrix

The instance of type submatrix.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_INVALID_BASETYPE

10.2.17 INSTANCEELEMENTITERATOR

INSTANCEELEMENTITERATOR_DESTROY

Purpose

Destroy the iterator and free the associated memory.

Parameters

None.

Java Calling Sequence

instElemIterator.destroy();

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENTITERATOR_GETCOUNT

Purpose

Get the total number of elements accessible by the iterator.

Parameters

None.

Java Calling Sequence

```
T_LONG instanceCount = instElemIterator.getCount();
```

Returns:

Return-Name:	instanceCount
Return-Type:	T_LONG

The number of elements accessible by the iterator.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENTITERATOR_NEXTN

Purpose

Get the next n elements from the sequence.

Parameters

how_many (Type = T_LONG)

The number of requested elements.

Java Calling Sequence

InstanceElement[] instElems = instElemIterator.nextN(how_many);

Returns:

Return-Name:	instElems
Return-Type:	InstanceElementSequence

The next n instance elements from the instance sequence.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENTITERATOR_NEXTONE

Purpose

Get the next element from the sequence.

Parameters

None.

Java Calling Sequence

InstanceElement instElem = instElemIterator.nextOne();

Returns:

Return-Name:	instElem
Return-Type:	InstanceElement

The next instance element from the instance sequence.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

INSTANCEELEMENTITERATOR_RESET

Purpose

Reset the pointer in the element sequence to the first element.

Parameters

None.

Java Calling Sequence

instElemIterator.reset();

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.18 MEASUREMENT

MEASUREMENT_CREATESMATLINK

Purpose

Create a submatrix link. The submatrix link is only valid in the current session. When the session is closed the submatrix link will be destroyed.

Parameters

None.

Java Calling Sequence

```
SMatLink smLink = measurement.createSMatLink();
```

Returns:

Return-Name:	smLink
Return-Type:	SMatLink

The new submatrix link.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

MEASUREMENT_GETSMATLINKS

Purpose

Get the list of the submatrix links .

Parameters

None.

Java Calling Sequence

```
SMatLink[] smLinks = measurement.getSMatLinks();
```

Returns:

Return-Name:	smLinks
Return-Type:	SMatLinkSequence

The available submatrix links.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

MEASUREMENT_GETVALUEMATRIX

Purpose

Get the value matrix of a measurement.

Parameters

None.

Java Calling Sequence

```
ValueMatrix vm = measurement.getValueMatrix();
```

Returns:

Return-Name:	vm
Return-Type:	ValueMatrix

The value matrix.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

MEASUREMENT_REMOVESMATLINK

Purpose

Remove a submatrix link.

Parameters

smLink (Type = SMatLink)

The submatrix link to be removed.

Java Calling Sequence

```
measurement.removeSMatLink(smLink);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_FOUND

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.19 NAMEITERATOR

NAMEITERATOR_DESTROY

Purpose

Destroy the iterator and free the associated memory.

Parameters

None.

Java Calling Sequence

nameIterator.destroy();

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NameIterator_getCount

Purpose

Get the total number of elements accessible by the iterator.

Parameters

None.

Java Calling Sequence

```
T_LONG nameCount = nameIterator.getCount();
```

Returns:

Return-Name:	nameCount
Return-Type:	T_LONG

The number of elements accessible by the iterator.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NAMEITERATOR_NEXTN

<u>Purpose</u>

Get the next n elements from the sequence.

<u>Parameters</u>

how_many (Type = T_LONG)

The number of requested elements.

<u>Java Calling Sequence</u>

Name[] names = nameIterator.nextN(how_many);

<u>Returns:</u>

Return-Name:	names
Return-Type:	NameSequence

The next n names from the name sequence.

<u>Possible exceptions (for details see section 10.3.14)</u>

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NAMEITERATOR_NEXTONE

Purpose

Get the next element from the sequence.

Parameters

None.

Java Calling Sequence

```
Name name = nameIterator.nextOne();
```

Returns:

Return-Name:	name
Return-Type:	Name

The next name from the name sequence.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NAMEITERATOR_RESET

Purpose

Reset the pointer in the element sequence to the first element.

Parameters

None.

Java Calling Sequence

```
nameIterator.reset();
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.20 NAMEVALUEITERATOR

NAMEVALUEITERATOR_DESTROY

Purpose

Destroy the iterator and free the associated memory.

Parameters

None.

Java Calling Sequence

```
nameValueIterator.destroy();
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NAMEVALUEITERATOR_GETCOUNT

Purpose

Get the total number of elements accessible by the iterator.

Parameters

None.

Java Calling Sequence

```
T_LONG nvCount = nameValueIterator.getCount();
```

Returns:

Return-Name:	nvCount
Return-Type:	T_LONG

The number of elements accessible by the iterator.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NameValueIterator_nextN

Purpose

Get the next n elements from the sequence.

Parameters

how_many (Type = T_LONG)

The number of requested elements.

Java Calling Sequence

NameValue[] nameValues = nameValueIterator.nextN(how_many);

Returns:

Return-Name:	nameValues
Return-Type:	NameValueSequence

The next n name-value pairs from the name-value pair sequence.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NAMEVALUEITERATOR_NEXTONE

Purpose

Get the next element from the sequence.

Parameters

None.

Java Calling Sequence

NameValue nameValue = nameValueIterator.nextOne();

Returns:

Return-Name:	nameValue
Return-Type:	NameValue

The next name-value pair from the name-value pair sequence.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NAMEVALUEITERATOR_RESET

Purpose

Reset the pointer in the element sequence to the first element.

Parameters

None.

Java Calling Sequence

```
nameValueIterator.reset();
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.21 NAMEVALUEUNITIDITERATOR

NAMEVALUEUNITIDITERATOR_DESTROY

Purpose

Destroy the iterator and free the associated memory.

Parameters

None.

Java Calling Sequence

```
nameValueUnitIdIterator.destroy();
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NameValueUnitIdIterator_getCount

Purpose

Get the total number of elements accessible by the iterator.

Parameters

None.

Java Calling Sequence

```
T_LONG nvuIdCount = nameValueUnitIdIterator.getCount();
```

Returns:

Return-Name:	nameValueUnitIdCount
Return-Type:	T_LONG

The number of elements accessible by the iterator.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NAMEVALUEUNITIDITERATOR_NEXTN

Purpose

Get the next n elements from the sequence.

Parameters

how_many (Type = T_LONG)

The number of requested elements.

Java Calling Sequence

```
NameValueUnitId[] nameValueUnitIds = nameValueUnitIdIterator.nextN(how_many);
```

Returns:

Return-Name:	nameValueUnitIdSeq
Return-Type:	NameValueSeqUnitId

The next n name-value-unit tuples from the name-value-unit tuple sequence.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NameValueUnitIdIterator_nextOne

Purpose

Get the next element from the sequence.

Parameters

None.

Java Calling Sequence

```
NameValueUnitId nameValueUnitId = nameValueUnitIdIterator.nextOne();
```

Returns:

Return-Name:	nameValueUnitId
Return-Type:	NameValueUnitId

The next name-value-unitId tuple from the name-value-unitId tuple sequence.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NAMEVALUEUNITIDITERATOR_RESET

Purpose

Reset the pointer in the element sequence to the first element.

Parameters

None.

Java Calling Sequence

nameValueUnitIdIterator.reset();

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.22 NameValueUnitIterator

NameValueUnitIterator_destroy

Purpose

Destroy the iterator and free the associated memory.

Parameters

None.

Java Calling Sequence

```
nameValueUnitIterator.destroy();
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NAMEVALUEUNITITERATOR_GETCOUNT

Purpose

Get the total number of elements accessible by the iterator.

Parameters

None.

Java Calling Sequence

```
T_LONG nvuCount = nameValueUnitIterator.getCount();
```

Returns:

Return-Name:	nvuCount
Return-Type:	T_LONG

The number of elements accessible by the iterator.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NameValueUnitIterator_nextN

Purpose

Get the next n elements from the sequence.

Parameters

how_many (Type = T_LONG)

The number of requested elements.

Java Calling Sequence

```
NameValueUnit[] nameValueUnits = nameValueUnitIterator.nextN(how_many);
```

Returns:

Return-Name:	nameValueUnits
Return-Type:	NameValueUnitSequence

The next n name-value-unit tuples from the name-value-unit tuple sequence.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NAMEVALUEUNITITERATOR_NEXTONE

Purpose

Get the next element from the sequence.

Parameters

None.

Java Calling Sequence

```
NameValueUnit nameValueUnit = nameValueUnitIterator.nextOne();
```

Returns:

Return-Name:	nameValueUnit
Return-Type:	NameValueUnit

The next name-value-unit tuple from the name-value-unit tuple sequence.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NameValueUnitIterator_reset

Purpose

Reset the pointer in the element sequence to the first element.

Parameters

None.

Java Calling Sequence

nameValueUnitIterator.reset();

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.23 NAMEVALUEUNITSEQUENCEITERATOR

NAMEVALUEUNITSEQUENCEITERATOR_DESTROY

Purpose

Destroy the iterator and free the associated memory.

Parameters

None.

Java Calling Sequence

```
nameValueUnitSequenceIterator.destroy();
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NAMEVALUEUNITSEQUENCEITERATOR_GETCOUNT

Purpose

Get the total number of elements accessible by the iterator.

Parameters

None.

Java Calling Sequence

T_LONG nameValueUnitCount = nameValueUnitSequenceIterator.getCount();

Returns:

Return-Name:	nameValueUnitCount
Return-Type:	T_LONG

The number of elements accessible by the iterator.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NAMEVALUEUNITSEQUENCEITERATOR_NEXTN

Purpose

Get the next n elements from the sequence.

Parameters

how_many (Type = T_LONG)

The number of requested elements.

Java Calling Sequence

```
NameValueSeqUnit[] nameValueSeqUnits =
nameValueUnitSequenceIterator.nextN(how_many);
```

Returns:

Return-Name:	nameValueSeqUnits
Return-Type:	NameValueSeqUnitSequence

The next n values of the name-value-unit sequence. For each NameValuUnit the next n values are stored in the value sequence.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NameValueUnitSequenceIterator_nextOne

Purpose

Get the next element from the iterator.

Parameters

None.

Java Calling Sequence

NameValueSeqUnit nameValueSeqUnit = nameValueUnitSequenceIterator.nextOne();

Returns:

Return-Name:	nameValueSeqUnit
Return-Type:	NameValueSeqUnit

The next name-value-unit tuple sequence from the name-value-unit tuple.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

NAMEVALUEUNITSEQUENCEITERATOR_RESET

Purpose

Reset the pointer in the element iterator to the first element.

Parameters

None.

Java Calling Sequence

nameValueUnitSequenceIterator.reset();

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.24 QUERY

QUERY_EXECUTEQUERY

Purpose

Execute query in asynchronous mode.

Parameters

params (Type = NameValueSequence)

Sequence of parameter names and values.The following parameter should be passed:

- Name: "QueryResultType";

  Type: ResultType.

  Comment: Specifies what kind of result is expected by the client, this parameter is required if the parameters isn't given at the method prepareQuery or the method createQuery of the interface QueryEvaluator.

  Default value: INSTELEM_ITERATOR_AS_RESULT
- Name: "Synchronous";

  Type: T_BOOLEAN

  Comment: In case of "true" guarantees synchronous execution.

  Default value: "false"

Java Calling Sequence

query.executeQuery(params);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_QUERY_PROCESSING_ERROR

AO_QUERY_INVALID_RESULTTYPE

QUERY_GETINSTANCES

Purpose

Get the query result. This method should only be called after the query has been executed. It returns an iterator on the instances that were found by the query.

Parameters

None.

Java Calling Sequence

```
InstanceElementIterator result = query.getInstances();
```

Returns:

Return-Name:	result
Return-Type:	InstanceElementIterator

The result of the query as an instance element iterator.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_QUERY_PROCESSING_ERROR

AO_QUERY_TIMEOUT_EXCEEDED

AO_QUERY_INCOMPLETE

AO_QUERY_INVALID_RESULTTYPE

QUERY_GETQUERYEVALUATOR

Purpose

Get the QueryEvaluator object which is responsible for this query.

Parameters

None.

Java Calling Sequence

```
QueryEvaluator queryEvl = query.getQueryEvaluator();
```

Returns:

Return-Name:	queryEvl
Return-Type:	QueryEvaluator

The QueryEvaluator object which is responsible for this query.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
AO_TRANSACTION_NOT_ACTIVE

QUERY_GETSTATUS

Purpose

Return query status.

Returns INCOMPLETE if the query is still executing.

Returns COMPLETE if the query finished execution or if the query execution stopped because of an error or because the timeout was exceeded.

Parameters

None.

Java Calling Sequence

QueryStatus status = query.getStatus();

Returns:

Return-Name:	status
Return-Type:	QueryStatus

The status of the query.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

QUERY_GETTABLE

Purpose

Get the query result. This method should only be called after the query has been executed. It returns the result as a structure.

Parameters

None.

Java Calling Sequence

```
NameValueSeqUnit[] result = query.getTable();
```

Returns:

Return-Name:	result
Return-Type:	NameValueSeqUnitSequence

The result of the query as a name value sequence unit sequence. Each name value sequence unit tuple is one column of the table. The name value sequence unit sequence is the table. The result structure can be very huge.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_QUERY_PROCESSING_ERROR

AO_QUERY_TIMEOUT_EXCEEDED

AO_QUERY_INCOMPLETE

AO_QUERY_INVALID_RESULTTYPE

QUERY_GETTABLEROWS

Purpose

Get the query result. This method should only be called after the query has been executed. It returns an iterator on the name-value-unit sequence.

Parameters

None.

Java Calling Sequence

```
NameValueUnitSequenceIterator result = query.getTableRows();
```

Returns:

Return-Name:	result
Return-Type:	NameValueUnitSequenceIterator

The result of the query as a name value unit sequence iterator. Each name value unit tuple is one cell of the table. The name value unit sequence is one row of the table.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_QUERY_PROCESSING_ERROR

AO_QUERY_TIMEOUT_EXCEEDED

AO_QUERY_INCOMPLETE

AO_QUERY_INVALID_RESULTTYPE

QUERY_PREPAREQUERY

Purpose

Do the query pre-processing (optimization, etc.) Call can be omitted by the client. In this case the functionality is executed on the call of executeQuery.

Parameters

params (Type = NameValueSequence)

Sequence of parameter names and values. The following parameter should be passed:

- Name: "QueryResultType";

  Type: ResultType.

  Comment: Specifies what kind of result is expected by the client, this parameter is required if the parameter isn't given at the method createQuery of the interface QueryEvaluator.

  Default value: INSTELEM_ITERATOR_AS_RESULT

Java Calling Sequence

query.prepareQuery(params);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_QUERY_PROCESSING_ERROR

AO_QUERY_INVALID_RESULTTYPE

10.2.25 QUERYEVALUATOR

QUERYEVALUATOR_CREATEQUERY

Purpose

Create a query object to execute it in asynchronous mode.

Parameters

queryStr (Type = T_STRING)

The query string

params (Type = NameValueSequence)

Sequence of parameter names and values. The following parameters should be passed:

- Name: "QueryResultType";

  Type: ResultType.

  Comment: Specifies what kind of result is expected by the client.

  Default value: INSTELEM_ITERATOR_AS_RESULT

- Name: "MaxDuration";

  Type: T_LONG

  Comment: Can be used to restrict the processing time. The time is given in milliseconds,

  Default value: 0 (no restriction)

Java Calling Sequence

Query queryObj = queryEvaluator.createQuery(queryStr,params);

Returns:

Return-Name:	queryObj
Return-Type:	Query

The query object.

Possible exceptions (for details see section 10.3.14)

AO_QUERY_TYPE_INVALID

AO_QUERY_INVALID

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

QUERYEVALUATOR_GETINSTANCES

Purpose

Evaluate a query in synchronous mode.

Parameters

queryStr (Type = T_STRING)

The query string.

params (Type = NameValueSequence)

Sequence of parameter names and values. The following parameter should be passed:

- Name: "MaxDuration";

  Type: T_LONG

  Comment: Can be used to restrict the processing time. The time is given in milliseconds,

  Default value: 0 (no restriction)

Java Calling Sequence

InstanceElementIterator result = queryEvaluator.getInstances(queryStr,params);

Returns:

Return-Name:	result
Return-Type:	InstanceElementIterator

The result of the query as an instance element iterator.

Possible exceptions (for details see section 10.3.14)

AO_QUERY_TYPE_INVALID

AO_QUERY_INVALID

AO_QUERY_PROCESSING_ERROR

AO_QUERY_TIMEOUT_EXCEEDED

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

QUERYEVALUATOR_GETTABLE

Purpose

Evaluate a query in synchronous mode.

Parameters

queryStr (Type = T_STRING)

The query string.

params (Type = NameValueSequence)

Sequence of parameter names and values. The following parameters should be passed:

Name: "MaxDuration";

Type: T_LONG

Comment: Can be used to restrict the processing time. The time is given in milliseconds,

Default value: 0 (no restriction)

Java Calling Sequence

```
NameValueSeqUnit[] result = queryEvaluator.getTable(queryStr,params);
```

Returns:

Return-Name:	result
Return-Type:	NameValueSeqUnitSequence

The result of the query as a name value sequence unit sequence. Each name value sequecne unit tuple is one column of the table. The name value sequence unit sequence is the table. The result structure can be very huge.

Possible exceptions (for details see section 10.3.14)

AO_QUERY_TYPE_INVALID

AO_QUERY_INVALID

AO_QUERY_PROCESSING_ERROR

AO_QUERY_TIMEOUT_EXCEEDED

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

QUERYEVALUATOR_GETTABLEROWS

Purpose

Evaluate a query in synchronous mode.

Parameters

queryStr (Type = T_STRING)

The query string.

params (Type = NameValueSequence)

Sequence of parameter names and values. The following parameter should be passed:

- Name: "MaxDuration";

  Type: T_LONG

  Comment: Can be used to restrict the processing time. The time is given in milliseconds,

  Default value: 0 (no restriction)

Java Calling Sequence

NameValueUnitSequenceIterator result = queryEvaluator.getTableRows(queryStr,params);

Returns:

Return-Name:	result
Return-Type:	NameValueUnitSequenceIterator

The result of the query as a name value unit sequence iterator. Each name value unit tuple is one cell of the table. The name value unit sequence is one row of the table.

Possible exceptions (for details see section 10.3.14)

AO_QUERY_TYPE_INVALID

AO_QUERY_INVALID

AO_QUERY_PROCESSING_ERROR

AO_QUERY_TIMEOUT_EXCEEDED

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.26 SMATLINK

SMATLINK_GETLINKTYPE

Purpose

Get the link or build type.

Parameters

None.

Java Calling Sequence

BuildUpFunction linkType = sMatLink.getLinkType();

Returns:

Return-Name:	linkType
Return-Type:	BuildUpFunction

The link type.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

SMatLink_getOrdinalNumber

Purpose

Get the ordinal or sequence number

Parameters

None.

Java Calling Sequence

```
T_LONG ordinalNumber = sMatLink.getOrdinalNumber();
```

Returns:

Return-Name:	ordinalNumber
Return-Type:	T_LONG

The sequence number.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

SMATLINK_GETSMAT1

Purpose

Get the first submatrix of the link.

Parameters

None.

Java Calling Sequence

```
SubMatrix subMatrix = sMatLink.getSMat1();
```

Returns:

Return-Name:	subMatrix
Return-Type:	SubMatrix

The first submatrix of the link.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

SMatLink_getSMat1Columns

Purpose

Get the bind columns of the first submatrix used in the link (e.g. Time).

Parameters

None.

Java Calling Sequence

```
Column[] columns = sMatLink.getSMat1Columns();
```

Returns:

Return-Name:	columns
Return-Type:	ColumnSequence

The columns of the first submatrix.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

SMATLINK_GETSMAT2

Purpose

Get the second submatrix of the link.

Parameters

None.

Java Calling Sequence

SubMatrix subMatrix = sMatLink.getSMat2();

Returns:

Return-Name:	subMatrix
Return-Type:	SubMatrix

The second submatrix of the link.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

SMatLink_getSMat2Columns

Purpose

Get the bind columns of the second submatrix used in the link (e.g. Time).

Parameters

None.

Java Calling Sequence

```
Column[] columns = sMatLink.getSMat2Columns();
```

Returns:

Return-Name:	columns
Return-Type:	ColumnSequence

The columns of the second submatrix.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

SMATLINK_SETLINKTYPE

Purpose

Set the build or link type.

Parameters

linkType (Type = BuildUpFunction)

The requested build-up function.

Java Calling Sequence

```
sMatLink.setLinkType(linkType);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_BUILDUP_FUNCTION

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

SMatLink_setOrdinalNumber

Purpose

Set the ordinal or sequence number.

Parameters

ordinalNumber (Type = T_LONG)

The sequence number.

Java Calling Sequence

sMatLink.setOrdinalNumber(ordinalNumber);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_ORDINALNUMBER

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

SMATLINK_SETSMAT1

Purpose

Set the first submatrix of the link.

Parameters

subMat1 (Type = SubMatrix)

The first submatrix of the submatrix link.

Java Calling Sequence

```
sMatLink.setSMat1(subMatrix);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_SUBMATRIX

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

SMATLINK_SETSMAT1COLUMNS

Purpose

Set the bind columns of the first submatrix used in the link (e.g. Time).

Parameters

columns (Type = ColumnSequence)

The column sequence of the submatrix.

Java Calling Sequence

```
sMatLink.setSMat1Columns(columns);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_COLUMN

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

SMATLINK_SETSMAT2

Purpose

Set the second submatrix of the link.

Parameters

subMat2 (Type = SubMatrix)

The second submatrix of the submatrix link.

Java Calling Sequence

```
sMatLink.setSMat2(subMatrix);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_INVALID_SUBMATRIX

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

SMatLink_setSMat2Columns

Purpose

Set the bind columns of the second submatrix used in the link (e.g. Time). If there is more than one column bound the column sequence must be identical with the column sequence of the first submatrix.

Parameters

columns (Type = ColumnSequence)

The column sequence of the submatrix.

Java Calling Sequence

sMatLink.setSMat2Columns(columns);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_INVALID_COLUMN

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.27 SUBMATRIX

SUBMATRIX_GETCOLUMNS

Purpose

Get the columns of the submatrix. The column is not inherited from the InstanceElement interface. This is the only way to get a column. The columns are used in the SMatLink interface to build the value matrix. The pattern is case sensitive and may contain wildcard characters.

Parameters

colPattern (Type = Pattern)

The name or the search pattern for the column names.

Java Calling Sequence

```
Column[] columns = subMatrix.getColumns(colPattern);
```

Returns:

Return-Name:	columns
Return-Type:	ColumnSequence

The columns of the submatrix.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

SUBMATRIX_GETVALUEMATRIX

Purpose

Get a value matrix of the submatrix.

Parameters

None.

Java Calling Sequence

```
ValueMatrix valueMatrix = subMatrix.getValueMatrix();
```

Returns:

Return-Name:	valueMatrix
Return-Type:	ValueMatrix

The value matrix.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_SMATLINK

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

SUBMATRIX_LISTCOLUMNS

Purpose

Get the names of the columns of the submatrix. The name sequence is identical with the names of the related instances. The pattern is case sensitive and may contain wildcard characters.

Parameters

colPattern (Type = Pattern)

The name or the search pattern for the column names.

Java Calling Sequence

```
Name[] columnNames = subMatrix.listColumns(colPattern);
```

Returns:

Return-Name:	columnNames
Return-Type:	NameSequence

The column names of the submatrix.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.2.28 VALUEMATRIX

VALUEMATRIX_ADDCOLUMN

Purpose

Add a column to the value matrix. It is only allowed to create a value vector if the value matrix is created from a submatrix.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The data is made permanent on transaction commit. Until the transaction is committed or until the access to the measurement point of the value matrix it is allowed to have different number of values in the different value vectors. After the new column is added, it is possible to set the values of the column.

Parameters

newColumn (Type = NameUnit)

The name and unit of the column to add.

Java Calling Sequence

valueMatrix.addColumn(newColumn);

Returns:

Return-Name:	column
Return-Type:	Column

The new column.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_NAME

AO_INVALID_SET_TYPE

AO_IS_MEASUREMENT_MATRIX

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

VALUEMATRIX_ADDCOLUMNSCALEDBY

Purpose

Add a column to the value matrix. It is only allowed to create a value vector if the value matrix is created from a submatrix. The column will be scaled by the given scaling column.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The data is made permanent on transaction commit. Until the transaction is committed or until the access to the measurement point of the value matrix it is allowed to have different number of values in the different value vectors. After the new column is added, it is possible to set the values of the column.

Parameters

newColumn (Type = NameUnit)

The name and unit of the column to add.

scalingColumn (Type = Column)

The scaling column.

Java Calling Sequence

```
Column column = valueMatrix.addColumnScaledBy(newColumn,scalingColumn);
```

Returns:

Return-Name:	column
Return-Type:	Column

The new column.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_NAME

AO_INVALID_SET_TYPE

AO_IS_MEASUREMENT_MATRIX

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_NO_SCALING_COLUMN

VALUEMATRIX_DESTROY

Purpose

Destroy the object on the server. This method is used to tell the server that this object is not used anymore by the client. Access to this object after the destroy method will lead to an exception.

Parameters

None.

Java Calling Sequence

```
vmObj.destroy()
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_TRANSACTION_NOT_ACTIVE

VALUEMATRIX_GETCOLUMNCOUNT

Purpose

Get the column count of the value matrix.

Parameters

None.

Java Calling Sequence

```
T_LONG columnCount = valueMatrix.getColumnCount();
```

Returns:

Return-Name:	columnCount
Return-Type:	T_LONG

The number of columns of the value matrix.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

VALUEMATRIX_GETCOLUMNS

Purpose

Get the columns of the value matrix no matter whether the column is dependent or independent. The pattern is case sensitive and may contain wildcard characters.

Parameters

colPattern (Type = Pattern)

The name or the search pattern for the column names.

Java Calling Sequence

```
Column[] columns = valueMatrix.getColumns(colPattern);
```

Returns:

Return-Name:	columns
Return-Type:	ColumnSequence

The columns of the value matrix, no matter whether the column is dependent, independent or scaled by another one

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

VALUEMATRIX_GETCOLUMNSSCALEDBY

Purpose

Get the columns which are scaled by the given column.

Parameters

scalingColumn (Type = Column)

The scaling column.

Java Calling Sequence

```
Column[] columns = valueMatrix.getColumnsScaledBy(scalingColumn);
```

Returns:

Return-Name:	columns
Return-Type:	ColumnSequence

The columns which are scaled by the given input column.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_NO_SCALING_COLUMN

VALUEMATRIX_GETINDEPENDENTCOLUMNS

Purpose

Get the independent columns of the value matrix. The independent columns are the columns used to build the value matrix.

Parameters

colPattern (Type = Pattern)

The name or the search pattern for the independent column name.

Java Calling Sequence

```
Column[] columns = valueMatrix.getIndependentColumns(colPattern);
```

Returns:

Return-Name:	columns
Return-Type:	ColumnSequence

The independent column of the value matrix.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

VALUEMATRIX_GETROWCOUNT

Purpose

Get the row count of the value matrix.

Parameters

None.

Java Calling Sequence

T_LONG rowCount = valueMatrix.getRowCount();

Returns:

Return-Name: rowCount

Return-Type: T_LONG

The number of rows of the value matrix.

Possible exceptions (for details see section 10.3.14)

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

VALUEMATRIX_GETSCALINGCOLUMNS

Purpose

Get the scaling column of the value matrix.

Parameters

colPattern (Type = Pattern)

The name or the search pattern for the scaling column name.

Java Calling Sequence

```
Column[] columns = valueMatrix.getScalingColumns(colPattern);
```

Returns:

Return-Name:	columns
Return-Type:	ColumnSequence

The scaling columns of the value matrix.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

VALUEMATRIX_GETVALUE

Purpose

Get the values of different columns of the value matrix.

Parameters

columns (Type = ColumnSequence)

The requested columns.

startPoint (Type = T_LONG)

The starting point in the column.

count (Type = T_LONG)

The number of points to be retrieved. 0 mean until end of column.

Java Calling Sequence

```
Column columns[] = vmObj.getColumns("*");
T LONG startPoint = 0;
T LONG count = 100;
NameValueSeqUnit values[] = vmObj.getValue(columns, startPoint, count);
```

Returns:

Return-Name:	values
Return-Type:	NameValueSeqUnitSequence

The values of the different columns. The name of the return structure corresponds with the name of the column. The unit corresponds with the unit of the column. The order of the result might not match the order in the request sequence.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

VALUEMATRIX_GETVALUEMEAPOINT

Purpose

Get a measurement point of the value matrix. The parameter meaPoint specifies the row of the matrix. The iterator allows to access all elements in the row.

Parameters

meaPoint (Type = T_LONG)

The measurement point.

Java Calling Sequence

```
NameValueUnitIterator valueMeaPoint = valueMatrix.getValueMeaPoint(meaPoint);
```

Returns:

Return-Name:	valueMeaPoint
Return-Type:	NameValueUnitIterator

The requested measurement point.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

VALUEMATRIX_GETVALUEVECTOR

Purpose

Get the values or a part of values of the column from the value matrix. The parameter column specifies from which column the values will be returned. The startPoint and pointCount specify the part of the vector. A startPoint = 0 and pointCount = rowCount will return the entire vector. When startPoint >= rowCount an exception is thrown. If startPoint + pointCount > rowCount only the remaining values of the vector are returned and no exception is thrown. Use the getName and getUnit method of the interface column for the name and the unit of the column. The name and the value are not stored at each element of the vector. The return type TS_ValueSeq is not a sequence of TS_Value but a special structure.

Parameters

col (Type = Column)

The column to retrieve the values from.

startPoint (Type = T_LONG)

The starting point in the column.

count (Type = T_LONG)

The number of points to be retrieved.

Java Calling Sequence

```
TS_Value[] valueVector = valueMatrix.getValueVector(col,startPoint,count);
```

Returns:

Return-Name:	valueVector
Return-Type:	TS_ValueSeq

The requested column values of the value matrix.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_COLUMN

AO_INVLAID_COUNT

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

VALUEMATRIX_LISTCOLUMNS

Purpose

Get the names of the columns of the value matrix no matter whether the column is dependent or independent. The pattern is case sensitive and may contain wildcard characters.

Parameters

colPattern (Type = Pattern)

The name or the search pattern for the column names.

Java Calling Sequence

```
Name[] columnNames = valueMatrix.listColumns(colPattern);
```

Returns:

Return-Name:	columnNames
Return-Type:	NameSequence

The column names of the value matrix, no matter whether the column is dependent, independent or scaled by another one.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

VALUEMATRIX_LISTCOLUMNSSCALEDBY

Purpose

List the names of the columns which are scaled by the given column.

Parameters

scalingColumn (Type = Column)

The scaling column.

Java Calling Sequence

```
Name[] columnNames = valueMatrix.listColumnsScaledBy(scalingColumn);
```

Returns:

Return-Name:	columnNames
Return-Type:	NameSequence

The names of the columns which are scaled by the given input column.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

AO_NO_SCALING_COLUMN

VALUEMATRIX_LISTINDEPENDENTCOLUMNS

Purpose

Get the names of the independent columns of the value matrix. The independent columns are the columns used to build the value matrix.

Parameters

colPattern (Type = Pattern)

The name or the search pattern for the independent column name.

Java Calling Sequence

```
Name[] columnNames = valueMatrix.listIndependentColumns(colPattern);
```

Returns:

Return-Name:	columnNames
Return-Type:	NameSequence

The names of the independent columns of the value matrix.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

VALUEMATRIX_LISTSCALINGCOLUMNS

Purpose

Get the names of the scaling columns of the value matrix.

Parameters

colPattern (Type = Pattern)

The name or the search pattern for the scaling column name.

Java Calling Sequence

```
Name[] columnNames = valueMatrix.listScalingColumns(colPattern);
```

Returns:

Return-Name:	columnNames
Return-Type:	NameSequence

The names of the scaling columns of the value matrix.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

VALUEMATRIX_REMOVEVALUEMEAPOINT

Purpose

Remove the values of the columns at a given measurement point. Remove the number of points of the given column. If the count is 0 all points until the end of the column are removed.

Parameters

columnNames (Type = NameSequence)

The columns from which the measurement points are to be removed.

meaPoint (Type = T_LONG)

The measurement point to be removed.

count (Type = T_LONG)

The number of points to be removed from each column.

Java Calling Sequence

```
valueMatrix.removeValueMeaPoint(columnNames,meaPoint,count);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_COUNT

AO_NOT_FOUND

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

VALUEMATRIX_REMOVEVALUEVECTOR

Purpose

Remove the values from a value vector. Beginning at startPoint the number of values specified in count are removed. If count is 0 all values from the startPoint until the end of the vector are removed.

Parameters

col (Type = Column)

The column from which the values are to be removed.

startPoint (Type = T_LONG)

The starting point for the value removal.

count (Type = T_LONG)

The number of points to be removed from the column.

Java Calling Sequence

```
valueMatrix.removeValueVector(col,startPoint,count);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_COLUMN

AO_INVALID_COUNT

AO_NOT_FOUND

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

VALUEMATRIX_SETVALUE

Purpose

Create or modify a number of value vectors in a value matrix.

It is only allowed to create a value vector if the value matrix is created from a submatrix.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The data is made permanent on transaction commit. Until the transaction is committed or until the access to the measurement point of the value matrix it is allowed to have different numbers of values in the different value vectors.

The names of the parameter values are the names of the columns. The values are the new values of the column. (setValueMeaPoint allows only one point, the TS_ValueSeq allows more then one point) There is a sequence of name value pairs (setValueVector allows only one column), so a block of values can be modified.

The names of the columns have to exist.

When a client creates a ValueMatrix based on AoMeasurement this methods behaves as follows:

The server checks the submatrices and creates all necessary instances of AoSubmatrix, AoLocalColumn and AoMeasurementQuantity. The values of the name attribute of AoSubmatrix must be generated by the server. The value will be equal to the value of the attribute ID (converted to DT_STRING). Missing instances of AoMeasurementQuantity will be created by the server, too.

The mandatory attributes will get the following default values:
- Name supplied by client
- Datatype copied from AoQuantity.default_datatype
- Typesize copied from AoQuantity.default_typesize
- Interpolation no interpolation
- Rank copied from AoQuantity.default_rank
- Dimension copied from AoQuantity.default_dimension

The server takes the value of the channel (supplied by client) and looks up the corresponding instance in AoQuantity using the attribute default_meq_name.Parameters

set (Type = SetType)

The set type. It may be one of

INSERT: Insert the values from startPoint, the current available values from startPoint are moved to the end of the new inserted values.

APPEND: The value of startPoint is ignored, the values are appended at the end of the current values.

UPDATE: Update or modify the values from startPoint, the current values are overwritten. If the number of values is greater than the number of values in the vector, the measurement point is automatically appended.

REMOVE: Remove the number of values from each column, starting with startPoint. The name of the column is given as the name of the NameValueSeqUnit, the number of values to remove is given in the number of the values in the value.

startPoint (Type = T_LONG)

The measurement point.

value (Type = NameValueSeqUnitSequence)

The values to be inserted.

Java Calling Sequence

valueMatrix.setValue(set,startPoint,value);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_COLUMN

AO_INVALID_SET_TYPE

AO_IS_MEASUREMENT_MATRIX

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

VALUEMATRIX_SETVALUEMEAPOINT

Purpose

Create or modify a measurement point of a value matrix. The sequence of name values specifies the names of the column with the new valuesThe names of the columns have to exist.

When a client creates a ValueMatrix based on AoMeasurement this methods behaves as follows:

The server checks the submatrices and creates all necessary instances of AoSubmatrix, AoLocalColumn and AoMeasurementQuantity. The values of the name attribute of AoSubmatrix must be generated by the server. The value will be equal to the value of the attribute ID (converted to DT_STRING). Missing instances of AoMeasurementQuantity will be created by the server, too. The mandatory attributes will get the following default values:
- Name supplied by client
- Datatype copied from AoQuantity.default_datatype
- Typesize copied from AoQuantity.default_typesize
- Interpolation no interpolation
- Rank copied from AoQuantity.default_rank
- Dimension copied from AoQuantity.default_dimension

The server takes the value of the channel (supplied by client) and looks up the corresponding instance in AoQuantity using the attribute default_meq_name.

Parameters

set (Type = SetType)

The set type. It may be one of

INSERT: Insert the values at meaPoint, the current values at meaPoint are moved to the end of the new inserted values.

APPEND: The value of meaPoint is ignored, the values are appended at the end of the current values.

UPDATE: Update or modify the values at meaPoint, the current values are overwritten. If meaPoint is bigger than the number of values in the vector, the measurement point is automatically appended.

meaPoint (Type = T_LONG)

The measurement point.

value (Type = NameValueSequence)

The values to be inserted.

Java Calling Sequence

```
valueMatrix.setValueMeaPoint(set,meaPoint,value);
```

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_IS_MEASUREMENT_MATRIX

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

VALUEMATRIX_SETVALUEVECTOR

Purpose

Create or modify a value vector in a value matrix.

It is allowed to modify the object outside a transaction but it is recommended to activate a transaction.

The data is made permanent on transaction commit. Until the transaction is committed or until the access to the measurement point of the value matrix it is allowed to have different number of values in the different value vectors.

When a client creates a ValueMatrix based on AoMeasurement this methods behaves as follows:

The server checks the submatrices and creates all necessary instances of AoSubmatrix, AoLocalColumn and AoMeasurementQuantity. The values of the name attribute of AoSubmatrix must be generated by the server. The value will be equal to the value of the attribute ID (converted to DT_STRING). Missing instances of AoMeasurementQuantity will be created by the server, too.

The mandatory attributes will get the following default values:
- Name supplied by client
- Datatype copied from AoQuantity.default_datatype
- Typesize copied from AoQuantity.default_typesize
- Interpolation no interpolation
- Rank copied from AoQuantity.default_rank
- Dimension copied from AoQuantity.default_dimension

The server takes the value of the channel (supplied by client) and looks up the corresponding instance in AoQuantity using the attribute default_meq_name.

Parameters

col (Type = Column)

The column whose values are to be set.

set (Type = SetType)

The set type. It may be one of

INSERT: Insert the values from startPoint, the current available values from startPoint are moved to the end of the new inserted values.

APPEND: The value of startPoint is ignored, the values are appended at the end of the current values.

UPDATE: Update or modify the values from startPoint, the current values are overwritten. If the number of values in the parameter values is too big the values are automatically appended.

startPoint (Type = T_LONG)

The starting point for the new values.

value (Type = TS_ValueSeq)

The values to be inserted.

Java Calling Sequence

valueMatrix.setValueVector(col,set,startPoint,value);

Returns:

None.

Possible exceptions (for details see section 10.3.14)

AO_BAD_PARAMETER

AO_CONNECTION_LOST

AO_IMPLEMENTATION_PROBLEM

AO_INVALID_COLUMN

AO_INVALID_SET_TYPE

AO_IS_MEASUREMENT_MATRIX

AO_NOT_IMPLEMENTED

AO_NO_MEMORY

AO_SESSION_NOT_ACTIVE

10.3 ASAM ODS Type Definitions

This section contains besides some general information on data types, the tables and descriptions for the ASAM ODS enumerations (like data types, constants, relations, exceptions, etc.) as well as the definitions for structures, unions etc..

10.3.1 ASAM ODS Data Types

The ASAM ODS data types of the OO-API are shown in the following table. In the meantime ASAM has harmonized the data types between all ASAM standards. Section 2.5 gives general information on data types in ASAM and ASAM ODS and describes the relationship between ASAM-wide harmonized data types and the legacy data types of ASAM ODS. Since there are several implementations available basing on the ASAM ODS data types, it was decided to keep them in the OO-API. However, anyone who intends to base on the harmonized ASAM data types according to section 2.5 may do so. Section 10.6 will provide an IDL-file which contains the mapping between the ASAM ODS data types and the harmonized ASAM data types.

The following table provides information on the data types used in the OO-API. While the names of the data types usually are mapped to primitive types in the specific context (Java, C++, ...), the names of the enumeration items are platform- and programming language-independent. Therefore the table lists these enumeration names together with a short description of the underlying data type. More information can be found in section 2.5.

data type enumeration name	Description
DT_UNKNOWN	Unknown data type
DT_STRING	String
DT_SHORT	Short value (integer, 16 bit)
DT_FLOAT	Float value (32 bit)
DT_BOOLEAN	Boolean value
DT_BYTE	Byte value (integer, 8 bit)
DT_LONG	Long value (integer, 32 bit)
DT_DOUBLE	Double precision float value (64 bit)
DT_LONGLONG	LongLong value (integer, 64 bit)
DT_COMPLEX	Complex value (32 bit each part)
DT_DCOMPLEX	Complex value (64 bit each part)
DT_ID	Id (64 bit); deprecated; use DT_LONGLONG
DT_DATE	Date, given as a string
DT_BYTESTR	Bytestream
DT_BLOB	Blob (Binary large object)
DS_STRING	String sequence
DS_SHORT	Short sequence
DS_FLOAT	Float sequence
DS_BOOLEAN	Boolean sequence
DS_BYTE	Byte sequence
DS_LONG	Long sequence

DS_DOUBLE	Double sequence
DS_LONGLONG	LongLong sequence
DS_COMPLEX	Complex sequence
DS_DCOMPLEX	Double complex sequence
DS_ID	Id sequence; deprecated; see above
DS_DATE	Date sequence
DS_BYTESTR	Bytestream sequence
DT_EXTERNALREFERENCE	External reference
DS_EXTERNALREFERENCE	Sequence of external reference
DT_ENUM	Enumeration
DS_ENUM	Sequence of enumerations

10.3.2 SUPPORTED DATA TYPE CONVERSIONS

Both, the ASAM ODS base model and the ASAM ODS API specifications include data types whose names may differ. Section 2.5.2 describes the relationship between the API data types and the base model data types.

In the base model, all base attributes have strictly defined data types. Normally the application attributes derived from the base attributes must have the same data type. There are some data types which are compatible, so ASAM ODS will allow the designer of the application model to overload the data type of the base attributes. The allowed alternatives to the attribute data types given in the base model are described in section 2.5.3.All base attributes not shown in that section are only allowed with the default data types defined in the base model.

The ASAM ODS server and client have to take care that the attributes may have a different data type than defined in the base model.

A similar situation exists with base relations: their relation ranges are specified in the base model with a few allowed exceptions, further explained in section 2.5.4.

10.3.3 ASAM ODS CONSTANTS

Constant	Value	Name	Description	DataType enum name
Lock Mode	0	LOCK_INSTANCEELEMENT	Lock the instance element. (Default LockMode)	DT_SHORT
	1	LOCK_APPLICATIONELEMENT	Lock the application element, all instances of the application element are locked.	DT_SHORT
	2	LOCK_CHILDREN	Lock the children of the locked object. This mode can be combined with one of the upper two modes.	DT_SHORT
QueryConstants	0	MaxDurationDEFAULT	Default value of max duration parameter of the query (no limitations).	DT_LONG
	"MaxDuration"	MaxDuration	The ASAM ODS max duration parameter of the query.	DT_STRING
	"QueryResultType"	QueryResultType	The ASAM ODS query result type parameter.	DT_STRING
	ResultType::INSTELEM_ITERATOR_AS_RESULT	QueryResultTypeDEFAULT	Default value of the ASAM ODS query result type parameter.	DT_SHORT
ResultType	0	INSTELEM_ITERATOR_AS_RESULT	Iterator of instance elements as result of the query (the default).	DT_SHORT
	1	TABLE_ITERATOR_AS_RESULT	Iterator for table access as result type of the query.	DT_SHORT
	2	TABLE_AS_RESULT	Table as result type of the query.	DT_SHORT
SecurityLevel	0	NO_SECURITY	No security defined.	DT_SHORT
	2	INSTANCE_SECURITY	Security scaled for instance elements.	DT_SHORT
	4	ATTRIBUTE_SECURITY	Security scaled for application attributes.	DT_SHORT
	1	ELEMENT_SECURITY	Security scaled for the application element.	DT_SHORT
SecurityRights	1	SEC_READ	Read access is allowed.	DT_SHORT
	2	SEC_WRITE	Write access is allowed.	DT_SHORT
	4	SEC_UPDATE	Update access to an existing object is allowed.	DT_SHORT
	8	SEC_DELETE	Delete of the object is allowed.	DT_SHORT
	16	SEC_GRANT	Grant right is allowed.	DT_SHORT

10.3.4 ASAM ODS ATTRIBUTE TYPES

Enumeration	SortKey	Name	Description
AttrType	0	APPLATTR_ONLY	Report only application attributes.
	1	INSTATTR_ONLY	Report only instance attributes.
	2	ALL	All attributes.

10.3.5 ASAM ODS BUILD UP FUNCTIONS

Enumeration	SortKey	Name	Description
BuildUpFunction	0	BUP_JOIN	Join the columns
	1	BUP_MERGE	Merge the columns
	2	BUP_SORT	Sort the columns

10.3.6 ASAM ODS QUERY STATUS

Enumeration	SortKey	Name	Description
QueryStatus	0	COMPLETE	The execution is ready.
	1	INCOMPLETE	The execution is still running.

10.3.7 ASAM ODS RIGHTS SET

Enumeration	SortKey	Name	Description
RightsSet	0	SET_RIGHT	Set the given rights, overwrite the existing rights.
	1	ADD_RIGHT	Add the given rights to the existing rights.
	2	REMOVE_RIGHT	Remove the given rights form the existing rights.

10.3.8 ASAM ODS Select Operation Code

SelOpCode gives query instructions like "equal", "greater" etc. So far, these arguments were case sensitive. There was a demand to add these arguments also for case insensitive comparison operations. Therefore, the SelOpCodes for case insensitivity were added. These arguments have the prefix “CI_”.

Enumeration	SortKey	Name	Description
SelOpcode	0	EQ	Equal
	1	NEQ	Not equal
	2	LT	Less then
	3	GT	Greater than
	4	LTE	Less then equal
	5	GTE	Greater then equal
	6	INSET	In set, value can be a sequence.
	7	NOTINSET	Not in set, value can be a sequence.
	8	LIKE	Like, use pattern matching, see Pattern for the wildcard definitions.
	9	CI_EQ	Equal (Case insensitive)
	10	CI_NEQ	Not equal (Case insensitive)
	11	CI_LT	Less then (Case insensitive)
	12	CI_GT	Greater than (Case insensitive)
	13	CI_LTE	Less then equal (Case insensitive)
	14	CI_GTE	Greater then equal (Case insensitive)
	15	CI_INSET	In set, value can be a sequence. (Case insensitive)
	16	CI_NOTINSET	Not in set, value can be a sequence. (Case insensitive)
	17	CI_LIKE	Like, use pattern matching, see Pattern for the wildcard definitions. (Case insensitive)

10.3.9 ASAM ODS Select Operator

Enumeration	SortKey	Name	Description
SelOperator	0	AND	AND the two conditions.
	1	OR	OR the two conditions.
	2	NOT	Negate the next condition.
	3	OPEN	Open brackets.
	4	CLOSE	Close brackets

10.3.10 ASAM ODS SET TYPE

Enumeration	SortKey	Name	Description
SetType	0	APPEND	Append data to the value matrix.
	1	INSERT	Insert data into the value matrix.
	2	UPDATE	Modify existing data of the value matrix.
	3	REMOVE	Remove the given information.

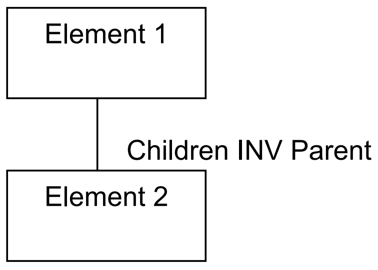
10.3.11 ASAM ODS SEVERITY FLAG

Enumeration	SortKey	Name	Description
SeverityFlag	0	SUCCESS	Ok.
	1	INFORMATION	Information.
	2	WARNING	Warning.
	3	ERROR	Error.

10.3.12 ASAM ODS RELATIONS

A relation is the connection between two ASAM ODS elements in both directions. There is no inverse relation, the ASAM ODS OO-API allows asking for the inverse name of the relation but it is only one relation. The get- and set- methods are defined from the first element, the getInverse- and setInverse- methods work from the second element.

<u>Example</u>



GetRelationName returns "Children"

GetInverseRelationName returns "Parent"

GetElem1 will return 'Element 1'

GetElem2 will return 'Element 2'

When a relation is created between two elements the inverse way is also created and when a relation is removed the inverse way is also removed. This is valid for the application model and the instances.

When a new relation is created care must be taken that the correct 'Element 1' and 'Element 2' is given at the corresponding set function.

THE RELATION_TYPE

There are three types of the relations:

FATHER_CHILD	there is a hierarchical relation between two elements. The FATHER_CHILD relation is used for the uniqueness of the instance when accessing it through the ASAM-PATH. (E.g. AoTest and AoSubTest)
INFO	there is an informational relation between the two elements. (E.g. AoMeasurementQuantity and AoQuantity)
INHERITANCE	there is an inheritance relation between two elements, the relation is only allowed between two elements of the same base type.

THE RELATIONSHIP

Besides the type of relation we have also the relationships. If we like to ask for the related elements we need the relationship and not the relation type. The following relationships are defined:

FATHER	The father element is requested of the relations of the type FATHER_CHILD. The father element is 'element 1' of the relation.
CHILD	The child element is requested of the relations of the type FATHER_CHILD. The child element is the 'element 2' of the relation.
INFO_TO	The target element is requested of the relations of the type INFO, the target element is the 'element 2' of relation.
INFO_FROM	The source element is requested of the relations of the type INFO, the target element is the 'element 1' of relation.
INFO_REL	The direction of the informational relation does not matter, so all elements connected with an informational relation are reported.
SUPER	The super element of the inheritance relation is given. The super element is the 'element 1' of the relation.
SUB	The sub element of the inheritance relation is given, the sub element is the 'element 2' of the relation.
ALL_REL	All related elements of all kind (types) of relation are reported.

The method CreateRelation and RemoveRelation of the interface InstanceElement is able to recognize the order which application element of the instance elements are the 'element 1' or 'element 2' of the application relation.

THE RELATION_RANGE

The relation range gives the allowed number of relations from one element to the other element of the relation. The following relation ranges are defined:

Min is the minimum number of relations. (0 or 1)

Max is the maximum number of relations (1 or MANY).

For the relation range the methods getRelationRange, getInverseRelationRange, setRelationRange and setInverseRelationRange were introduced.

Using this RELATION_RANGE the Obligatory flag is no longer needed at the relation, so the corresponding methods on the interfaces BaseRelation and ApplicationRelation are also no longer needed.

If there is a base relation, the relation type and the relation range of the base relation are used for the application relation. If there is no base relation the default relation type will be INFO with a relationship INFO_TO. Depending on the data type of the attribute the relation range is set. A data type enumeration name of DT_* means a relation range of 0:1. A name of DS_* means a relation range of 0:MANY.

The relation type is given in the following table.

Relation type	Value	Meaning
FATHER_CHILD	0	Father-child relation.
INFO	1	Informational relation.
INHERITANCE	2	Inheritance relation.

The relationship is given below.

Relationship	Value	Meaning
FATHER	0	Father
CHILD	1	Child
INFO_TO	2	Directed informational relationship.
INFO_FROM	3	Directed informational relationship.
INFO_REL	4	Informational relationship (no direction)
SUPERTYPE	5	Inheritance relationship: super type.
SUBTYPE	6	Inheritance relationship: subtype.
ALL_REL	7	Any of the relationships above.

The relation range has the following structure:

```
typedef struct RELATIONRANGE RelationRange;
struct RELATIONRANGE {
   Int2     min;
   Int2     max;
};
```

where:

min	The minimum number of relations; one of {-2, 0, 1}. If the minimum value is 1 there must always a reference be set. The value -2 means not initialized.
max	The maximum number of relation; one of {-2, -1, 1}. The value -1 means MANY (that is: There is no specified maximum; any amount greater or equal to the minimum is allowed.) The value -2 means not initialized.

10.3.13 ASAM ODS PATTERNS

A “Pattern” is always case sensitive.

A “Pattern” may have wildcard characters as follows:

- The wildcard character “?” for one (1) matching character is the question mark.
- The wildcard character “*” for a sequence of matching characters is the asterisk.
- The wildcard character ESCAPE *\\*.

10.3.14 ASAM ODS EXCEPTIONS

The ASAM ODS OO-API methods raise exceptions (AoException) whenever a problem is detected or an error is encountered. The reason for the exception is given as a character string. The reason string contains one of the reasons listed below. If additional information is provided, the reason is terminated by a colon (“:”). After the colon any implementation specific information may follow. The reasons are listed in alphabetical order.

Example for a typical exception reason with implementation specific information:

“NetworkFailure: RPC timeout encountered, node Jupiter, port number 553654711”

Without the implementation dependent information the reason string would be:

“NetworkFailure”

The following table lists all exceptions and their respective descriptions.

Exception	Description
`AO_ACCESS_DENIED`	The remote server denied the access. If this error occurred it was not even possible to present the authentication information. This means the authentication information might be correct but the server refused the access already at a lower level.
`AO_BAD_OPERATION`	The BAD_OPERATION error code is used when a method is invalid in a marshalling operation.
`AO_BAD_PARAMETER`	A parameter of the wrong type was passed to the method. The minorCode tells which parameter (1,

Exception	Description
	2, 3 or 4) is bad. If more than one parameter is bad, only the first one is reported. This error can occur only in non-typesave language bindings. Those language bindings also impose the problem that not all parameter errors are automatically detectable.
AO_CONNECTION_LOST	Due to hardware or network software problem the connection to the server was lost.
AO_CONNECT_FAILED	The connect to a server failed. This error may occur if the server is down or currently unreachable.
AO_CONNECT_REFUSED	The connection was refused by the server. This error may occur if the presented authentication information is either incorrect or incomplete. This error shall not occur if the server does not accept any more sessions due to overload problems. See AO_SESSION_LIMIT_REACHED for this case.
AO_DUPLICATE_BASE_ATTRIBUTE	Any application element may have only one base attribute of a certain type. This means it may have only one attribute of base attribute type NAME, one ID, one VERSION and so on.
AO_DUPLICATE_NAME	The implicitly or explicitly specified name is already in use but it is required to be unique.
AO_DUPLICATE_VALUE	The attribute is marked unique in the application model. Thus duplicate values are not allowed.
AO_HAS_BASE_ATTRIBUTE	Base attribute found. It is not allowed to modify the data type, the unique flag and obligatory flag.
AO_HAS_BASE_RELATION	Base relation found. It is not allowed to modify the relation type, relation range or relationship of an application relation derived from a base relation.
AO_HAS_INSTANCES	The operation is not allowed for elements that have instances.
AO_HAS_REFERENCES	The requested operation is not permitted because the target element has references.
AO_IMPLEMENTATION_PROBLEM	This error is reserved for the reporting of implementation-specific problems that are not properly handled by the standard error definitions. An application should not crash if this error occurs but there is no way to react to this error other than reporting and ignoring it. The intention of this error is not to leave implementation-specific errors unreported.
AO_INCOMPATIBLE_UNITS	The units are incompatible. No conversion rule is known.

Exception	Description
AO_INVALID_ASAM_PATH	The specified ASAM path is invalid.
AO_INVALID_ATTRIBUTE_TYPE	The requested attribute type is invalid.
AO_INVALID_BASETYPE	The specified base type is invalid.
AO_INVALID_BASE_ELEMENT	The base element is invalid in this context. If this is an element of type AoMeasurement, another element of this type may already exist.
AO_INVALID_BUILDUP_FUNCTION	The specified build-up function is invalid.
AO_INVALID_COLUMN	The specified column is invalid.
AO_INVALID_COUNT	The specified number of points is invalid (probably negative).
AO_INVALID_DATATYPE	The data type is not allowed in the given context or it conflicts with an existing data type definition. This error may also occur in non-typesave language bindings. To avoid this error in all language bindings it is recommended to use always the definitions of the enumeration “DataType“.
AO_INVALID_ELEMENT	The element is invalid in this context.
AO_INVALID_LENGTH	The given length is invalid. Negative length values are not allowed.
AO_INVALID_ORDINALNUMBER	The ordinal number is either already used or less than zero.
AO_INVALID_RELATION	The relation is invalid. The related elements and the base relation do not fit.
AO_INVALID_RELATIONSHIP	This error may occur only in non-typesave language bindings. To avoid this error in all language bindings it is recommended to use always the definitions of the enumeration "Relationship".
AO_INVALID_RELATION_RANGE	The specified relation range is invalid.
AO_INVALID_RELATION_TYPE	This error may occur only in non-typesave language bindings. To avoid this error in all language bindings it is recommended to use always the definitions of the enumeration "RelationType".
AO_INVALID_SET_TYPE	The specified set-type is invalid.
AO_INVALID_SMATLINK	The submatrix link is invalid. Either submatrix 1 or 2 is not specified or the ordinal number is missing when there is more than one SMatLink.
AO_INVALID_SUBMATRIX	The specified submatrix is invalid.
AO_IS_BASE_ATTRIBUTE	The application attribute is of a base attribute type. It cannot be changed. If this is required, the application attribute has to be removed from its

Exception	Description
	application element and re-created. This error may occur if an application attribute derived from a base attribute a. shall be overwritten by a different base attribute type. b. shall receive a different data type. c. shall receive a different unique-flag. d. shall receive a different obligatory-flag.
AO_IS_BASE_RELATION	Properties of base relations may not be changed.
AO_IS_MEASUREMENT_MATRIX	The matrix is a complex, generated matrix from a measurement not just a simple submatrix. It is only allowed to modify submatrices but not the composed measurement matrices.
AO_MATH_ERROR	A computation error occurred. This can be an overflow, an underflow or a division by zero.
AO_MISSING_APPLICATION_ELEMENT	A required application element is missing.
AO_MISSING_ATTRIBUTE	A required (obligatory) attribute is missing.
AO_MISSING_RELATION	A required relation is missing.
AO_MISSING_VALUE	An obligatory value is missing (the AO_VF_DEFINED flag is zero).
AO_NOT_FOUND	The requested element was not found. This error occurs only in remove or rename operations if the subject of the operation is not found. All get- and list-methods return an empty list if the requested item is not found.
AO_NOT_IMPLEMENTED	The requested method is not yet implemented. This error is not allowed to occur in a certified implementation (except if a client accesses the Query interface; the Query interface is no official functionality of ASAM ODS Version 5.0; a server must throw this exception if that interface is accessed by a client). It is intended to allow partial operational tests during the development process.
AO_NOT_UNIQUE	This error occurs if the instances of a property are required to be unique.
AO_NO_MEMORY	No more volatile memory available.
AO_NO_PATH_TO_ELEMENT	A free-floating element was detected. No navigation path leads to this element.
AO_NO_SCALING_COLUMN	The column is no scaling column.
AO_OPEN_MODE_NOT_SUPPORTED	The requested open mode is not supported. Valid open modes are "read" and "write". Anything else is rejected with this error and no session is

Exception	Description
	created.
AO_QUERY_INCOMPLETE	The execution of the query was not yet completed.
AO_QUERY_INVALID	Some error in the query string or some inconsistency between the return type of the query string and the result type specified by parameter "QueryResultType".
AO_QUERY_INVALID_RESULTTYPE	The requested result type of the query does not match with the previous definition of the result type.
AO_QUERY_PROCESSING_ERROR	Some error occurred during the execution of the query.
AO_QUERY_TIMEOUT_EXCEEDED	It was not possible to execute the query within the time limit set by parameter "MaxDuration".
AO_QUERY_TYPE_INVALID	The server does not support the specified query language type.
AO_SESSION_LIMIT_REACHED	The server does not accept any new connections. This error may occur if the server reached the session limit for a distinct user or the total number of sessions allowed.
AO_SESSION_NOT_ACTIVE	The session is no longer active. This error occurs if an attempt is made to call a method of a closed session. This error shall not be confused with the error AO_CONNECTION_LOST.
AO_TRANSACTION_ALREADY_ACTIVE	There may be only one active transaction at one time. If this error occurs there is already an active transaction. That transaction remains active in case of this error.
AO_TRANSACTION_NOT_ACTIVE	Write operations have to be done always in the context of a transaction. This error occurs if no transaction is active during a write operation.
AO_UNKNOWN_ERROR	Use the zero as unknown error to avoid confusing error messages if no error code has been set.
AO_UNKNOWN_UNIT	The unit is unknown.

10.3.15 ASAM ODS Specific Types

Type	Data Type	Defined Type	Description
Scalar	String	T_STRING	
Scalar	Boolean	T_BOOLEAN	
Scalar	Short	T_SHORT	

Type	Data Type	Defined Type	Description
Scalar	Float	T_FLOAT	
Scalar	Octet	T_BYTE	
Scalar	Long	T_LONG	
Scalar	Double	T_DOUBLE	
Scalar	T_STRING	Name	
Scalar	T_STRING	Pattern	Wildcard character for one character is '?'. Wildcard character for more than one character is '*'.
Scalar	T_STRING	BaseType	
Scalar	T_STRING	T_DATE	
Scalar	Blob	T_BLOB	
Sequence	T_BYTE	T_BYTESTR	
Sequence	T_BOOLEAN	S_BOOLEAN	
Sequence	T_BYTE	S_BYTE	
Sequence	T_DOUBLE	S_DOUBLE	
Sequence	T_FLOAT	S_FLOAT	
Sequence	T_LONG	S_LONG	
Sequence	T_SHORT	S_SHORT	
Sequence	T_STRING	S_STRING	
Sequence	T_COMPLEX	S_COMPLEX	
Sequence	T_DCOMPLEX	S_DCOMPLEX	
Sequence	T_LONGLONG	S_LONGLONG	
Sequence	BaseType	BaseTypeSequence	
Sequence	Name	NameSequence	
Sequence	NameValue	NameValueSequence	
Sequence	NameValueUnit	NameValueUnitSequence	
Sequence	Column	ColumnSequence	
Sequence	SmatLink	SMatLinkSequence	
Sequence	SubMatrix	SubMatrixSequence	
Sequence	T_ID	S_ID	
Sequence	T_DATE	S_DATE	
Sequence	T_BYTESTR	S_BYTESTR	
Sequence	S_STRING	SS_STRING	
Sequence	S_SHORT	SS_SHORT	
Sequence	S_FLOAT	SS_FLOAT	

Type	Data Type	Defined Type	Description
Sequence	S_BOOLEAN	SS_BOOLEAN	
Sequence	S_BYTE	SS_BYTE	
Sequence	S_LONG	SS_LONG	
Sequence	S_DOUBLE	SS_DOUBLE	
Sequence	S_LONGLONG	SS_LONGLONG	
Sequence	S_COMPLEX	SS_COMPLEX	
Sequence	S_DCOMPLEX	SS_DCOMPLEX	
Sequence	S_ID	SS_ID	
Sequence	S_DATE	SS_DATE	
Sequence	S_BYTESTR	SS_BYTESTR	
Sequence	T_BLOB	S_BLOB	
Scalar	T_LONGLONG	T_ID	
Sequence	ApplicationElement	ApplicationElementSequence	
Sequence	ApplicationRelation	ApplicationRelationSequence	
Sequence	ApplicationAttribute	ApplicationAttributeSequence	
Sequence	BaseRelation	BaseRelationSequence	
Sequence	BaseAttribute	BaseAttributeSequence	
Sequence	BaseElement	BaseElementSequence	
Sequence	InstanceElement	InstanceElementSequence	
Sequence	NameValueSeqUnit	NameValueSeqUnitSequence	
Sequence	T_ExternalReference	S_ExternalReference	
Sequence	S_ExternalReference	SS_ExternalReference	
Sequence	ApplAttr	ApplAttrSequence	
Sequence	ApplElem	ApplElemSequence	Application element definition sequence. The application elements are given with the method getElements() at the interface ApplicationStructure.
Sequence	ApplRel	ApplRelSequence	Application relation sequence. The application relations are given with the method getRelations()

Type	Data Type	Defined Type	Description
			at the interface ApplicationStructure.
Sequence	AIDName	AIDNameSequence	Sequence of AID - Name pairs.
Sequence	AIDNameUnitId	AIDNameUnitIdSequence	Sequence of AID - Name - UnitId tuple.
Sequence	AIDNameValueSeqUnitId	AIDNameValueSeqUnitId Sequence	Sequence of AID - Name - UnitId tuple.
Sequence	ElemId	ElemIdSequence	Sequence of element Id's.
Sequence	AttrResultSet	AttrResultSetSequence	Sequence of the attribute result sets.
Sequence	ElemResultSet	ElemResultSetSequence	Sequence of the element result sets.
Sequence	SelOrder	SelOrderSequence	The sequence of order criteria. The first criteria is the importance criteria.
Sequence	SelValue	SelValueSequence	Sequence of attribute search conditions.
Sequence	SelOperator	SelOperatorSequence	
Sequence	ACL	ACLSequence	The sequence of access control list entries for a requested object.

10.3.16 ASAM ODS UNIONS

UnionName	CaseName	DataType	Name
SelItem	SEL_OPERATOR_TYPE	SelOperator	operator
SelItem	SEL_VALUE_TYPE	SelValueExt	value
TS_Union	DT_STRING	T_STRING	stringVal
TS_Union	DT_SHORT	T_SHORT	shortVal
TS_Union	DT_FLOAT	T_FLOAT	floatVal
TS_Union	DT_BYTE	T_BYTE	byteVal
TS_Union	DT_BOOLEAN	T_BOOLEAN	booleanVal
TS_Union	DT_LONG	T_LONG	longVal
TS_Union	DT_DOUBLE	T_DOUBLE	doubleVal
TS_Union	DT_LONGLONG	T_LONGLONG	longlongVal
TS_Union	DT_COMPLEX	T_COMPLEX	complexVal

UnionName	CaseName	DataType	Name
TS_Union	DT_DCOMPLEX	T_DCOMPLEX	dcomplexVal
TS_Union	DT_ID	T_ID	idVal
TS_Union	DT_DATE	T_DATE	dateVal
TS_Union	DT_BYTESTR	T_BYTESTR	bytestrVal
TS_Union	DT_BLOB	T_BLOB	blobVal
TS_Union	DS_STRING	S_STRING	stringSeq
TS_Union	DS_SHORT	S_SHORT	shortSeq
TS_Union	DS_FLOAT	S_FLOAT	floatSeq
TS_Union	DS_BYTE	S_BYTE	byteSeq
TS_Union	DS_BOOLEAN	S_BOOLEAN	booleanSeq
TS_Union	DS_LONG	S_LONG	longSeq
TS_Union	DS_DOUBLE	S_DOUBLE	doubleSeq
TS_Union	DS_LONGLONG	S_LONGLONG	longlongSeq
TS_Union	DS_COMPLEX	S_COMPLEX	complexSeq
TS_Union	DS_DCOMPLEX	S_DCOMPLEX	dcomplexSeq
TS_Union	DS_ID	S_ID	idSeq
TS_Union	DS_DATE	S_DATE	dateSeq
TS_Union	DS_BYTESTR	S_BYTESTR	bytestrSeq
TS_Union	DT_EXTERNALREFERENCE	T_ExternalReference	extRefVal
TS_Union	DS_EXTERNALREFERENCE	S_ExternalReference	extRefSeq
TS_Union	DT_ENUM	T_LONG	enumVal
TS_Union	DS_ENUM	S_LONG	enumSeq
TS_UnionSeq	DT_STRING	S_STRING	stringVal
TS_UnionSeq	DT_SHORT	S_SHORT	shortVal
TS_UnionSeq	DT_FLOAT	S_FLOAT	floatVal
TS_UnionSeq	DT_BYTE	S_BYTE	byteVal
TS_UnionSeq	DT_BOOLEAN	S_BOOLEAN	booleanVal
TS_UnionSeq	DT_LONG	S_LONG	longVal
TS_UnionSeq	DT_DOUBLE	S_DOUBLE	doubleVal
TS_UnionSeq	DT_LONGLONG	S_LONGLONG	longlongVal
TS_UnionSeq	DT_COMPLEX	S_COMPLEX	complexVal
TS_UnionSeq	DT_DCOMPLEX	S_DCOMPLEX	dcomplexVal
TS_UnionSeq	DT_ID	S_ID	idVal
TS_UnionSeq	DT_DATE	S_DATE	dateVal
TS_UnionSeq	DT_BYTESTR	S_BYTESTR	bytestrVal
TS_UnionSeq	DT_BLOB	S_BLOB	blobVal
TS_UnionSeq	DS_STRING	SS_STRING	stringSeq
TS_UnionSeq	DS_SHORT	SS_SHORT	shortSeq
TS_UnionSeq	DS_FLOAT	SS_FLOAT	floatSeq

UnionName	CaseName	DataType	Name
TS_UnionSeq	DS_BYTE	SS_BYTE	byteSeq
TS_UnionSeq	DS_BOOLEAN	SS_BOOLEAN	booleanSeq
TS_UnionSeq	DS_LONG	SS_LONG	longSeq
TS_UnionSeq	DS_DOUBLE	SS_DOUBLE	doubleSeq
TS_UnionSeq	DS_LONGLONG	SS_LONGLONG	longlongSeq
TS_UnionSeq	DS_COMPLEX	SS_COMPLEX	complexSeq
TS_UnionSeq	DS_DCOMPLEX	SS_DCOMPLEX	dcomplexSeq
TS_UnionSeq	DS_ID	SS_ID	idSeq
TS_UnionSeq	DS_DATE	SS_DATE	dateSeq
TS_UnionSeq	DS_BYTESTR	SS_BYTESTR	bytestrSeq
TS_UnionSeq	DT_EXTERNALREFERENCE	S_ExternalReference	extRefVal
TS_UnionSeq	DS_EXTERNALREFERENCE	SS_ExternalReference	extRefSeq
TS_UnionSeq	DT_ENUM	S_LONG	enumVal
TS_UnionSeq	DS_ENUM	SS_LONG	enumSeq

10.3.17 ASAM ODS STRUCTURES

The following tables show the ASAM ODS structures. Wherever a data type is referenced that is represented in the DataType enumeration, its enumeration name is specified.

ACCESS CONTROL LIST (ACL)

Structure	DataType (resp. enum name)	Name	Description
ACL	DT_SHORT	rights	The access rights of the requested object.
	DT_LONGLONG	usergroupId	The usergroup Id.

ATTRIBUTE ID

Structure	DataType (resp. enum name)	Name	Description
AIDName	Name	aaName	The attribute, or measured quantity name.
	DT_LONGLONG	aid	The Id of the application element.
AIDNameUnitId	DT_LONGLONG	unitId	The unit of the attribute of the column. The unitId is the Id of instance element with the basetype AoUnit.
	AIDName	attr	The attribute of the application element (aid, name).
AIDNameValueSeqUnitId	DT_LONGLONG	unitId	The unit of the attribute of the column. The unitId is the Id of instance element with the basetype AoUnit.
	AIDName	attr	The attribute of the application element (aid, name).
	TS_ValueSeq	values	The column values with value flags.
AIDNameValueUnitId	DT_LONGLONG	unitId	The unit of the attribute of the column. The unitId is the Id of instance element with the basetype AoUnit.
	TS_Value	values	The attribute values with value flags.
	AIDName	attr	The attribute of the application element (aid, name).

APPLICATION ATTRIBUTE

Structure	DataType (resp. enum name)	Name	Description
ApplAttr	DT_BOOLEAN	isUnique	The indicator for unique attributes. The same boolean is returned by the method isUnique() of the interface ApplicationAttribute.
	DT_BOOLEAN	isObligatory	The indicator for mandatory attributes, the notNull indicator is set at the column of the table in the physical storage. The same boolean is returned at the method isObligatory() of the interface ApplicationAttribute.
	DT_LONG	length	The maximum possible length of values. The same length is returned by the method getLength() of the interface ApplicationAttribute.
	DataType	dType	The attribute data type. The same data type is given by the method getDataType of the interface ApplicationAttribute. At the RPC-API this information is stored in the field aDataType of the structure AttrSeq and the request AOP_GetAttr.
	Name	baName	The name of the base attribute, empty ("") if the application attribute is not derived from a base attribute. The same name is returned by the methods getName() of the BaseAttribute interface. The base attribute is given by the method getBaseAttribute() of the interface ApplicationAttribute. At the RPC-API this information is stored in the field aBName of the structure AttrSeq and the request AOP_GetAttr.
	Name	aaName	The application attribute name. The same name is returned by the method getName() of the ApplicationAttribute interface. At the RPC-API this information is stored in the field aAName of the structure AttrSeq and the request AOP_GetAttr.
	DT_LONGLONG	unitId	Id of the unit if global defined. The same Id is returned by the method getUnit() of the interface ApplicationAttribute. At the RPC-API this information is stored in the field aUnit of the structure AttrSeq and the request AOP_GetAttr.

Application Element

Structure	DataType (resp. enum name)	Name	Description
ApplElem	Name	beName	The base element name. All application elements are related to a base element. The same name is returned by the methods getName() of the BaseElement interface. The base element is given with the method getBaseElement() at the interface ApplicationElement. At the RPC-API this information is not delivered but the corresponding Id of the base element is stored in the field aiBId of the structure ApplInfSeq and the request AOP_GetApplInf.
	Name	aeName	The application element name. The name is given also with the method getName() at the interface ApplicationElement. At the RPC-API this information is stored in the field aiName of the structure ApplInfSeq and the request AOP_GetApplInf.
	DT_LONGLONG	aid	The application element id. The id is given also with the method getId() at the interface ApplicationElement. At the RPC-API this information is stored in the field aiAId of the structure ApplInfSeq and the request AOP_GetApplInf.
	ApplAttr Sequence	attributes	The attributes of application element. The attributes are given with the method getAttributes() of the interface ApplicationElement. There are no relations given in the this sequence.

Application Structure Value

Structure	DataType	Name	Description
ApplicationStructureValue	ApplRelSequence	applRels	The list of relations in the application model; the relation and the inverse relation are given.
	ApplElemSequence	applElems	The list of application elements.

APPLICATION RELATION

Structure	DataType (resp. enum name)	Name	Description
ApplRel	RelationRange	invRelationRange	The range of the inverse relation. The range of the relation is not stored in the physical storage. The inverse relation range is returned at the method getInverseRelationRange() of the interface ApplicationRelation.
		arRelationRange	The range of the relation. The range of the relation is not stored in the physical storage. The relation range is returned at the method getRelationRange() of the interface ApplicationRelation.
	RelationType	arRelationType	The type of the relation. The type of the relation is not stored in the physical storage. The relation type is returned at the method getRelationType() of the interface ApplicationRelation.
	Name	invBrName	Name of the inverse base relation from the elem2 to the elem1. The base relation is not stored in the physical storage.
	DT_LONGLONG	elem2	The target application element Id. The given Id is the Id of the application element returned from the method getElem2() of the interface ApplicationRelation. At the RPC-API this information is stored in the field arAid2 of the structure ApplRelSeq and the request AOP_GetApplInf.
	Name	brName	Name of the base relation from the elem1 to the elem2. The base relation is not stored in the physical storage.
	DT_LONGLONG	elem1	The source application element Id. The given Id is the Id of the application element returned from the method getElem1() of the interface ApplicationRelation. At the RPC-API this information is stored in the field arAid1 of the structure ApplRelSeq and the request AOP_GetApplInf.
	Name	invName	Name of the inverse relation. The name is returned with the method getInverseName() of the interface ApplicationRelation. The invName is not available in the physical storage for relational databases.
	Name	arName	The relation name. The name is returned with the method getName() of the interface ApplicationRelation. At the RPC-API this information is stored in the field arName of the structure ApplRelSeq and the request AOP_GetApplInf.

Attribute Result Set

Structure	DataType	Name	Description
AttrResultSet	NameValueSeqUnitId	attrValues	The first 'how_many' results. All values has the same AIDName and the same UnitId.
	NameValueUnitIdIterator	rest	The rest of the results.

Element ID

Structure	DataType enum name	Name	Description
ElemId	DT_LONGLONG	iid	The Id of the instance element.
		aid	The Id of the application element.

Element Result Set

Structure	DataType (resp. enum name)	Name	Description
ElemResultSet	DT_LONGLONG	aid	The Id of the application element.
	AttrResultSetSequence	attrValues	The selected attributes of the element. The number of values in each AttrResultSet are identical, the attributes of one element has always the position in the AttrResultSet.

Name Unit

Structure	DataType (resp. enum name)	Name	Description
NameUnit	DT_STRING	unit	Column unit as string.
	Name	valName	Attribute name or measured quantity name.

NAME VALUE

Structure	DataType (resp. enum name)	Name	Description
NameValue	TS_Value	value	Attribute value or column value (vector).
	Name	valName	Attribute name or measured quantity name.
NameValueSeqUnit	Name	valName	Column name or measured quantity name.
	DT_STRING	unit	Column unit as string.
	TS_ValueSeq	value	Column value (vector).
NameValueSeqUnitId	DT_LONGLONG	unitId	Column unit as Id.
	TS_ValueSeq	value	Column value (vector).
	Name	valName	Column name or measured quantity name.
NameValueUnit	TS_Value	value	Attribute value or column value (vector).
	DT_STRING	unit	Attribute or column unit as string.
	Name	valName	Attribute name or measured quantity name.
NameValueUnitId	Name	valName	Attribute name or measured quantity name.
	DT_LONGLONG	unitId	Id of attribute or column unit.
	TS_Value	value	Attribute value or column value (vector).

QUERY STRUCTURE

Structure	DataType	Name	Description
QueryStructure	SelValueSequence	condSeq	The query condition sequence. At the RPC-API this information is stored in the field nsSeq of the structure GetValReq and the request AOP_GetVal.
	SelOrderSequence	orderBy	The order by sequence. The order of the result set. At the RPC-API it is not possiable to set the order.
	Name	relName	Name of the relation. At the RPC-API this information is stored in the field refName of the structure GetValReq and the request AOP_GetVal.
	SelOperatorSequence	operSeq	The query condition operator sequence. At the RPC-API the operator is always ‚AND‘.
	AIDNameUnitIdSequence	anuSeq	The sequence of attributes to be reported. At the RPC-API this information is stored in the fields applId and nuSeq of the structure GetValReq and the request AOP_GetVal. At the RPC-API only one application element can be selected. A pattern is accepted for the attribute name.
	ElemId	relInst	The related instance. (aid == 0 && iid == 0) means no related instance specified. At the RPC-API this information is stored in the field elemId of the structure GetValReq and the request AOP_GetVal.

RELATION RANGE

Structure	DataType enum name	Name	Description
RelationRange	DT_SHORT	max	The maximum number in the range. -1 means MANY without a specified maximum number.
	DT_SHORT	min	The minimum number in the range.

SELECT ORDER

Structure	DataType (resp. enum name)	Name	Description
SelOrder	AIDName	attr	Attribute specification.
	DT_BOOLEAN	ascending	ascending order, FALSE means descending.

SELECT VALUE

Structure	DataType	Name	Description
SelValue	TS_Value	value	Value for the condition.
	SelOpcode	oper	The compare operator between the attribute and value.
	AIDNameValueUnitId	attr	The attribute specification with unit of the value.

STRUCTURE TYPES DEFINITION

Structure	DataType (resp. enum name)	Name	Description
T_COMPLEX	DT_FLOAT	i	The imaginary part of the complex number.
	DT_FLOAT	r	The real part of the complex number.
T_DCOMPLEX	DT_DOUBLE	i	The imaginary part of the double precision complex number.
	DT_DOUBLE	r	The real part of the double precision complex number.
T_ExternalReference	DT_STRING	location	Location of the external reference. (asam path or URL)
	DT_STRING	mimeType	MIME type of the external object.
	DT_STRING	description	Description of the external reference.
T_LONGLONG	DT_LONG	low	The least significant 32 bits of the 64 bit value.
	DT_LONG	high	The most significant 32 bits of the 64 bit value.
TS_Value	TS_Union	u	The value union for values of all known data types.
	DT_SHORT	flag	The value flags. For better efficiency, all flags of one value are encoded in one byte, which means, for each flag one bit will be used. For access to the single flags, constants for the bit masks are defined. The following value flags are supported: AO_VF_VALID (0x01) - the value is valid AO_VF_VISIBLE (0x02) - the value has to be visualized AO_VF_UNMODIFIED (0x04) - the value has not been modified AO_VF_DEFINED (0x08) - the value is defined (this flag is also used by the base layer to mark gaps in the value matrix)
TS_ValueSeq	TS_UnionSeq	u	The value union for values of all known data types.
	DS_SHORT	flag	See TS_Value, *flag*.

10.4 PROGRAMMING EXAMPLES

10.4.1 ACCESSING THE ASAM ODS FACTORY OBJECT VIA CORBA

The OO-API may be called by any interpreter or compiler language. The design of this application programmers interface has been done using CORBA-IDL. Thus, using the OO-API via CORBA is the most natural thing to do, and the following examples demonstrate the use of the OO-API with CORBA as mediator.

The beauty of this approach is the very small programming effort required to deploy CORBA. The only difference between using the OO-API with or without CORBA is the way to obtain the AoFactory object. For all other methods in the OO-API interfaces there is no difference between a call via CORBA and a local call – CORBA is transparent to the application programmer.

The requirements for CORBA services are limited to just the CORBA Name Service. The CORBA Name Service called “NameService” must be started. After this name service is running, an OO-API Object Request Broker (ORB) can register itself with the name service.

After that, any client can ask the CORBA name server for the registered OO-API ORB. As result the client receives the handle of an ASAM ODS Factory object (aoFactory) which can be used to establish an arbitrary number of sessions with the ASAM ODS server.

The example is written in Java but it would look very similar in any other programming language. In Java, the periods between the names are interpreted as delimiters between directory names. The paths are interpreted relative to the Java environment variable CLASSPATH. Another CLASSPATH means another implementation (which may be with or without CORBA).

The example does not contain any error checking or failure prevention code. The names of CORBA ORBs registered with CORBA name servers can be hierarchical similar to file names in directory structures. To make this first example not unnecessarily complicated the OO-API ORB in this example is named “serverName”. For more detailed information about CORBA Services and hierarchical name spaces see the appropriate CORBA documentation from the OMG (~~www.omg.org)~~.
Down here is an example how to get the AoFactory using the CORBA naming service. This example shows how to access the OO-API methods via CORBA.

```
// Import ASAM ODS factory classes.
import org.asam.ods.AoFactory;
import org.asam.ods.AoFactoryHelper;

// Import CORBA Name Service and ORB classes.
import org.omg.CosNaming.NameComponent;
import org.omg.CosNaming.NamingContext;
import org.omg.CosNaming.NamingContextHelper;
import org.omg.CORBA.ORB;

// Create and initialize the ORB.
```

```
ORB orb = ORB.init(args, null);

// Get the root naming context.
org.omg.CORBA.Object objRef =
orb.resolve_initial_references("NameService");
NamingContext ncRef = NamingContextHelper.narrow(objRef);

// Resolve the Object Reference in Naming.
NameComponent nc = new NameComponent("serverName", "serverType");
NameComponent path[] = {nc};
AoFactory aoFactory = AoFactoryHelper.narrow(ncRef.resolve(path));
```

10.5 OO-API DEFINITION FILE

On the following pages the ASAM ODS definition file is listed. There is one definition file for the OO-API: *ODS.IDL*. It describes the interfaces using CORBA IDL (interface definition language). All modules and functions of the definition file are explained in detail (purpose, calling sequence, returns, examples) in section 10.2.

10.5.1 ODS.IDL

```
//    ****************************************************
//    *  ASAM ODS Version 5.0 OO-API Interface Definition *
//    ****************************************************
//
//    Generated: Fri Mar 12 12:05:19 CET 2004
//    Last change: Fri Mar 12 2004
//
//    A list of ASAM ODS base elements can be found
//    in the description of AO_INVALID_BASE_TYPE.
//
//    The error descriptions are located at the end
//    of this file.
//
#ifndef ods_idl
#define ods_idl

module org {

/*
* ASAM services.
*/
module asam {

/*
* ASAM ODS service.
*/
module ods {
interface AoFactory;
interface AoSession;
interface ApplicationAttribute;
interface ApplicationElement;
interface ApplicationRelation;
interface ApplicationStructure;
interface BaseAttribute;
interface BaseElement;
interface BaseRelation;
interface BaseStructure;
interface Blob;
interface Column;
interface InstanceElement;
interface InstanceElementIterator;
interface Measurement;
interface NameIterator;
interface NameValueIterator;
interface NameValueUnitIterator;
interface SMatLink;
interface SubMatrix;
interface ValueMatrix;
interface NameValueUnitIdIterator;
interface ApplElemAccess;
interface QueryEvaluator;
interface Query;
interface NameValueUnitSequenceIterator;
interface EnumerationDefinition;
interface ElemResultSetExtSeqIterator;

/*
* The ASAM ODS error severity flags.
*/
enum SeverityFlag {
   SUCCESS,     // Ok.
   INFORMATION, // Information.
   WARNING,     // Warning.
   ERROR        // Error.
};

/*
* The ASAM ODS relation types.
*/
enum RelationType {
   FATHER_CHILD, // Father-child realation.
   INFO,         // Info relation.
   INHERITANCE   // Inheritance relation.
};

/*
* The ASAM ODS relationships.
*/
enum Relationship {
   FATHER,    // Father.
   CHILD,     // Child.
   INFO_TO,   // Directed informational relationship.
   INFO_FROM, // Directed informational relationship.
   INFO_REL,  // Informational relationship (no direction)
   SUPERTYPE, // Inheritance relationship: supertype.
   SUBTYPE,   // Inheritance relationship: subtype.
   ALL_REL    // Any of the relationships above.
};

/*
* The ASAM ODS data types.
*    DT_xxx  Basic data types.
*    DS_xxx  Sequence of basic data type.
*    ||
*    |+- T == Type, S == Sequences.
*    +-- D == Datatype.
*/
enum DataType {
   DT_UNKNOWN,            // Unknown datatype.
   DT_STRING,             // String.
   DT_SHORT,              // Short value (16 bit).
   DT_FLOAT,              // Float value (32 bit).
   DT_BOOLEAN,            // Boolean value.
   DT_BYTE,               // Byte value (8 bit).
   DT_LONG,               // Long value (32 bit).
   DT_DOUBLE,             // Double precision float value (64 bit).
   DT_LONGLONG,           // LongLong value (64 bit).
   DT_ID,                 // LongLong value (64 bit). Not used. DT_LONGLONG is
used instead.
   DT_DATE,               // Date.
   DT_BYTESTR,            // Bytestream.
   DT_BLOB,               // Blob.
   DT_COMPLEX,            // Complex value (32 bit each part).
   DT_DCOMPLEX,           // Complex value (64 bit each part).
   DS_STRING,             // String sequence.
   DS_SHORT,              // Short sequence.
   DS_FLOAT,              // Float sequence.
   DS_BOOLEAN,            // Boolean sequene.
   DS_BYTE,               // Byte sequence.
   DS_LONG,               // Long sequence.
   DS_DOUBLE,             // Double sequence.
   DS_LONGLONG,           // Longlong sequence.
   DS_COMPLEX,            // Complex sequence.
   DS_DCOMPLEX,           // Double complex sequence.
   DS_ID,                 // LongLong sequence. Not used. DS_LONGLONG is used
instead.
   DS_DATE,               // Date sequence.
   DS_BYTESTR,            // Bytestream sequence.
   DT_EXTERNALREFERENCE, // External reference.
   DS_EXTERNALREFERENCE, // Sequence of external reference.
   DT_ENUM,              // The enumeration datatype.
   DS_ENUM               // The enumeration sequence datatype.
};

/*
* The ASAM ODS build-up function codes for measurement views.
*/
enum BuildUpFunction {
   BUP_JOIN,  // Join the columns
   BUP_MERGE, // Merge the columns
   BUP_SORT   // Sort the columns
};

/*
* The ASAM ODS attribute type codes.
*/
enum AttrType {
   APPLATTR_ONLY, // Report only application attributes.
   INSTATTR_ONLY, // Report only instance attributes.
   ALL            // All attributes.
};

/*
* The ASAM ODS types for setting values.
*/
enum SetType {
   APPEND, // Append data to the value matrix.
   INSERT, // Insert data into the value matrix.
   UPDATE, // Modify existing data of the value matrix.
   REMOVE  // Remove the given information.
};

/*
* The ASAM ODS error codes.
*/
enum ErrorCode {
   AO_UNKNOWN_ERROR,
   AO_ACCESS_DENIED,
   AO_BAD_OPERATION,
   AO_BAD_PARAMETER,
   AO_CONNECT_FAILED,
   AO_CONNECT_REFUSED,
   AO_CONNECTION_LOST,
   AO_DUPLICATE_BASE_ATTRIBUTE,
   AO_DUPLICATE_NAME,
   AO_DUPLICATE_VALUE,
   AO_HAS_INSTANCES,
   AO_HAS_REFERENCES,
   AO_IMPLEMENTATION_PROBLEM,
   AO_INCOMPATIBLE_UNITS,
   AO_INVALID_ASAM_PATH,
   AO_INVALID_ATTRIBUTE_TYPE,
   AO_INVALID_BASE_ELEMENT,
   AO_INVALID_BASETYPE,
   AO_INVALID_BUILDUP_FUNCTION,
   AO_INVALID_COLUMN,
   AO_INVALID_COUNT,
   AO_INVALID_DATATYPE,
   AO_INVALID_ELEMENT,
   AO_INVALID_LENGTH,
   AO_INVALID_ORDINALNUMBER,
   AO_INVALID_RELATION,
   AO_INVALID_RELATION_RANGE,
   AO_INVALID_RELATION_TYPE,
   AO_INVALID_RELATIONSHIP,
   AO_INVALID_SET_TYPE,
   AO_INVALID_SMATLINK,
   AO_INVALID_SUBMATRIX,
   AO_IS_BASE_ATTRIBUTE,
   AO_IS_BASE_RELATION,
   AO_IS_MEASUREMENT_MATRIX,
   AO_MATH_ERROR,
   AO_MISSING_APPLICATION_ELEMENT,
   AO_MISSING_ATTRIBUTE,
   AO_MISSING_RELATION,
   AO_MISSING_VALUE,
   AO_NO_MEMORY,
   AO_NO_PATH_TO_ELEMENT,
   AO_NOT_FOUND,
   AO_NOT_IMPLEMENTED,
   AO_NOT_UNIQUE,
   AO_OPEN_MODE_NOT_SUPPORTED,
   AO_SESSION_LIMIT_REACHED,
   AO_SESSION_NOT_ACTIVE,
   AO_TRANSACTION_ALREADY_ACTIVE,
   AO_TRANSACTION_NOT_ACTIVE,
   AO_HAS_BASE_RELATION,
   AO_HAS_BASE_ATTRIBUTE,
   AO_UNKNOWN_UNIT,
   AO_NO_SCALING_COLUMN,
   AO_QUERY_TYPE_INVALID,
   AO_QUERY_INVALID,
   AO_QUERY_PROCESSING_ERROR,
   AO_QUERY_TIMEOUT_EXCEEDED,
   AO_QUERY_INCOMPLETE,
   AO_QUERY_INVALID_RESULTTYPE
};

/*
* The selection operators. SelOpcode gives query instructions
* like "equal", "greater" etc. So far, these arguments were
* case sensitive. There was a demand to add these arguments
* also for case insensitive comparison operations. Therefore,
* the SelOpcodes for case insensitivity were added. These
* arguments have the prefix "CI_".
*/
enum SelOpcode {
   EQ,           // Equal
   NEQ,          // Not equal
   LT,           // Less then
   GT,           // Greater then
   LTE,          // Less then equal
   GTE,          // Greater then equal
   INSET,        // In set, value can be a sequence.
   NOTINSET,     // Not in set, value can be a sequence.
   LIKE,         // like, use  pattern matching, see Pattern for the wildcard
definitions.
   CI_EQ,        // Equal. case insensitive for DT_STRING.
   CI_NEQ,       // Not equal. case insensitive for DT_STRING.
   CI_LT,        // Less then. case insensitive for DT_STRING.
   CI_GT,        // Greater then. case insensitive for DT_STRING.
   CI_LTE,       // Less then equal. case insensitive for DT_STRING.
   CI_GTE,       // Greater then equal. case insensitive for DT_STRING.
   CI_INSET,     // In set, value can be a sequence. case insensitive for
DT_STRING.
   CI_NOTINSET, // Not in set, value can be a sequence. case insensitive for
DT_STRING.
   CI_LIKE      // like, use  pattern matching, see Pattern for the wildcard
definitions. case insensitive for DT_STRING.
};

/*
* Operator, bracket open and close.
*/
enum SelOperator {
   AND,   // AND the two conditions.
   OR,    // OR the two conditions.
   NOT,   // Negate the next condition.
   OPEN,  // Open brackets.
   CLOSE  // Close brackets
};

/*
* The ASAM ODS types for setting access rights.
*/
enum RightsSet {
   SET_RIGHT,    // Set the given rights, overwrite the existing rights.
   ADD_RIGHT,    // Add the given rights to the existing rights.
   REMOVE_RIGHT  // Remove the given rights form the existing rights.
};

/*
* Status of the query execution.
*/
enum QueryStatus {
   COMPLETE,   // The execution is ready.
   INCOMPLETE  // The execution is still running.
};

/*
* The supported aggregate functiosn of the GetInstanceExt.
*/
enum AggrFunc {
   NONE,   // No aggregate function is used for attribute.
   COUNT,  // Count
   DCOUNT, // Distinct count
   MIN,    // Min; only for numerical values
   MAX,    // Max; only for numerical values
   AVG,    // Average; only for numerical values
   STDDEV  // Standard deviation; only for numerical values
};

/*
* The selection type.
*/
enum SelType {
   SEL_VALUE_TYPE,    // Selection value.
   SEL_OPERATOR_TYPE  // Selection logical operator.
};

/*
* The type of the join.
```

```
*/
enum JoinType {
   JTDEFAULT, // Force inner join.
   JTOUTER    // Force outer join on destination AID.
};

// Datatype definitions (T_xxx).
typedef string     T_STRING;
typedef boolean    T_BOOLEAN;
typedef short      T_SHORT;
typedef float      T_FLOAT;
typedef octet      T_BYTE;
typedef long       T_LONG;
typedef double     T_DOUBLE;
typedef T_STRING   Name;
typedef T_STRING   Pattern;
typedef T_STRING   BaseType;
typedef T_STRING   T_DATE;
typedef Blob       T_BLOB;

// Sequence definitions (S_xxx).
typedef sequence<T_BYTE> T_BYTESTR;
typedef sequence<T_BOOLEAN> S_BOOLEAN;
typedef sequence<T_BYTE> S_BYTE;
typedef sequence<T_DOUBLE> S_DOUBLE;
typedef sequence<T_FLOAT> S_FLOAT;
typedef sequence<T_LONG> S_LONG;
typedef sequence<T_SHORT> S_SHORT;
typedef sequence<T_STRING> S_STRING;
typedef sequence<BaseType> BaseTypeSequence;
typedef sequence<Name> NameSequence;
typedef sequence<Column> ColumnSequence;
typedef sequence<SMatLink> SMatLinkSequence;
typedef sequence<SubMatrix> SubMatrixSequence;
typedef sequence<T_DATE> S_DATE;
typedef sequence<T_BYTESTR> S_BYTESTR;
typedef sequence<S_STRING> SS_STRING;
typedef sequence<S_SHORT> SS_SHORT;
typedef sequence<S_FLOAT> SS_FLOAT;
typedef sequence<S_BOOLEAN> SS_BOOLEAN;
typedef sequence<S_BYTE> SS_BYTE;
typedef sequence<S_LONG> SS_LONG;
typedef sequence<S_DOUBLE> SS_DOUBLE;
typedef sequence<S_DATE> SS_DATE;
typedef sequence<S_BYTESTR> SS_BYTESTR;
typedef sequence<T_BLOB> S_BLOB;
typedef sequence<ApplicationElement> ApplicationElementSequence;
typedef sequence<ApplicationRelation> ApplicationRelationSequence;
typedef sequence<ApplicationAttribute> ApplicationAttributeSequence;
typedef sequence<BaseRelation> BaseRelationSequence;
typedef sequence<BaseAttribute> BaseAttributeSequence;
typedef sequence<BaseElement> BaseElementSequence;
typedef sequence<InstanceElement> InstanceElementSequence;
typedef sequence<SelOperator> SelOperatorSequence;

/*
* The ASAM ODS relation range structure.
*/
struct RelationRange {
   T_SHORT min; // The minimum number in the range.
   T_SHORT max; // The maximum number in the range. -1 means
                // MANY without a specified maximum number.
};

/*
* The ASAM ODS 64 bit integer structure. This type is
* represented in the datatype enumeration by DT_LONGLONG.
*/
struct T_LONGLONG {
   T_LONG high; // The most significant 32 bits of the 64 bit value.
   T_LONG low;  // The least significant 32 bits of the 64 bit value.
};

/*
* The ASAM ODS complex data structure. This type is
* represented in the datatype enumeration by DT_COMPLEX.
*/
struct T_COMPLEX {
   T_FLOAT r; // The real part of the complex number.
   T_FLOAT i; // The imaginary part of the complex number.
};

/*
* The ASAM ODS double-precision complex data structure. This
* type is represented in the datatype enumeration by DT_DCOMPLEX.
*/
struct T_DCOMPLEX {
   T_DOUBLE r; // The real part of the double precision complex number.
   T_DOUBLE i; // The imaginary part of the double precision complex
               // number.
};
```

```
/*
* The ASAM ODS name-unit tuple structure.
*/
struct NameUnit {
   Name     valName; // Attribute name or measured quantity name.
   T_STRING unit;    // Column unit as string.
};

/*
* The description of a reference object, the reference object
* can be an internal ASAM ODS object or an external object.
* This type is represented in the datatype enumeration by
* DT_EXTERNALREFERENCE.
*/
struct T_ExternalReference {
   T_STRING description; // Description of the external reference.
   T_STRING mimeType;    // Mime type of the external object.
   T_STRING location;    // Location of the external reference.
                         // (asam path or URL)
};

/*
* The application attribute information (metadata) definition.
* The same information is available at the interface
* ApplicationAttribute
*/
struct ApplAttr {
   Name       aaName;       // The application attribute name. The same
                            // name is returned by the method getName()
                            // of the ApplicationAttribute interface. At
                            // the protocol level 3 this information was
                            // stored in the field aAName of the
                            // structure AttrSeq and the request
                            // AOP_GetAttr.
   Name       baName;       // The name of the base attribute, empty ("")
                            // if the application attribute is not
                            // derived from a base attribute. The same
                            // name is returned by the methods getName()
                            // of the BaseAttribute interface. The base
                            // attribute is given by the method
                            // getBaseAttribute() of the interface
                            // ApplicationAttribute. At the protocol
                            // level 3 this information was stored in the
                            // field aBName of the structure AttrSeq and
                            // the request AOP_GetAttr.
   DataType   dType;        // The attribute data type. The same data
```

```
                             // type is given by the method getDataType of
                             // the interface ApplicationAttribute. At the
                             // protocol level 3 this information was
                             // stored in the field aDataType of the
                             // structure AttrSeq and the request
                             // AOP_GetAttr.
   T_LONG     length;        // The maximum possible length of values. The
                             // same length is returned by the method
                             // getLength() of the interface
                             // ApplicationAttribute.
   T_BOOLEAN  isObligatory;  // The indicator for mandatory attributes,
                             // the notNull indicator is set at the column
                             // of the table in the physical storage. The
                             // same boolean is returned at the method
                             // isObligatory() of the interface
                             // ApplicationAttribute. I think the
                             // information must be stored in the svcattr
                             // table and not in the table description of
                             // the database.
   T_BOOLEAN  isUnique;      // The indicator for unique attributes. The
                             // same boolean is returned by the method
                             // isUnique() of the interface
                             // ApplicationAttribute. I think the
                             // information must be stored in the svcattr
                             // table and not in the table description of
                             // the database.
   T_LONGLONG unitId;        // Id of the unit if global defined. The same
                             // Id is returned by the method getUnit() of
                             // the interface ApplicationAttribute. At the
                             // protocol level 3 this information was
                             // stored in the field aUnit of the structure
                             // AttrSeq and the request AOP_GetAttr.
};

/*
* The application relation info structure. The same information
* is available at the interface ApplicationRelation.
*/
struct ApplRel {
   T_LONGLONG    elem1;             // The source application element Id.
                                    // The given Id is the Id of the
                                    // application element returned from
                                    // the method getElem1() of the
                                    // interface ApplicationRelation. At
                                    // the protocol level 3 this
                                    // information was stored in the field
```

```
                                // arAid1 of the structure ApplRelSeq
                                // and the request AOP_GetApplInf.
T_LONGLONG    elem2;            // The target application element Id.
                                // The given Id is the Id of the
                                // application element returned from
                                // the method getElem2() of the
                                // interface ApplicationRelation. At
                                // the protocol level 3 this
                                // information was stored in the field
                                // arAid2 of the structure ApplRelSeq
                                // and the request AOP_GetApplInf.
Name          arName;           // The relation name. The name is
                                // returned with the method getName() of
                                // the interface ApplicationRelation.
                                // At the protocol level 3 this
                                // information was stored in the field
                                // arName of the structure ApplRelSeq
                                // and the request AOP_GetApplInf.
Name          invName;          // Name of the inverse relation. The
                                // name is return with the method
                                // getInverseName() of the interface
                                // ApplicationRelation. The invName is
                                // not available in the physical
                                // storage for relation databases.
Name          brName;           // Name of the base relation from the
                                // elem1 to the elem2. The base
                                // relation is also not stored in the
                                // physical storage.
Name          invBrName;        // Name of the inverse base relation
                                // from the elem2 to the elem1. The
                                // base relation is also not stored in
                                // the physical storage.
RelationType  arRelationType;   // The type of the relation. Type of
                                // the relation is not stored in the
                                // physical storage. The relation type
                                // is returned at the method
                                // getRelationType() of the interface
                                // ApplicationRelation.
RelationRange arRelationRange;  // The range of the relation. Range of
                                // the relation is not stored in the
                                // physical storage. The relation
                                // range is returned at the method
                                // getRelationRange() of the interface
                                // ApplicationRelation.
RelationRange invRelationRange; // The inverse range of the relation.
                                // Range of the relation is not stored
```

```
                                      // in the physical storage. The
                                      // inverse relation range is returned at
                                      // the method
                                      // getInverseRelationRange() of the
                                      // interface ApplicationRelation.
};

/*
* AID - Name pair.
*/
struct AIDName {
   T_LONGLONG aid;    // The Id of the application element.
   Name       aaName; // The attribute, or measured quantity name.
};

/*
* Instance element Id. The unique description of an instance
* element.
*/
struct ElemId {
   T_LONGLONG aid; // The Id of the application element.
   T_LONGLONG iid; // The Id of the instance element.
};

/*
* AID - Name - UnitId tuple.
*/
struct AIDNameUnitId {
   AIDName    attr;   // The attribute  of the application element (aid,
                      // name).
   T_LONGLONG unitId; // The unit of the attribute ot the column. The
                      // unitId is the Id of instance element with the
                      // basetype AoUnit.
};

/*
* Order criteria.
*/
struct SelOrder {
   AIDName   attr;      // Attribute specification.
   T_BOOLEAN ascending; // ascending order, FALSE means descending.
};

/*
* The access control list entry.
*/
```

```
struct ACL {
   T_LONGLONG usergroupId; // The usergroup Id.
   T_LONG     rights;      // The access rights of the requested object.
};

/*
* An initial right.
*/
struct InitialRight {
   T_LONG     rights;      // The initial access rights of the requested
                           // object.
   T_LONGLONG usergroupId; // The usergroup Id of the Initial right list.
   T_LONGLONG refAid;      // The referencing application element.
};

/*
* it’s quite the same sequence as in the QueryStructure of
* GetInstances with one exception. It has one new attribute
* called function, which is of the type AggrFunc. Thereby  it
* is possible to define aggregate functions on attribute level,
* without the need to parse the attribute name for a known
* aggregate function name. The default value of that attribute
* function is NONE, it symbolizes that no aggregate function
* should be applied on that attribute.If an aggregate function
* is used, it is also required to define a GroupSequence.It is
* also defined that a ‘*’ as attribute name, delivers all
* attributes of an element.
*/
struct SelAIDNameUnitId {
   AIDName    attr;      // The attribute  of the application element
                         // (aid, name).
   T_LONGLONG unitId;    // The unit of the attribute ot the column. The
                         // unitId is the Id of instance element with the
                         // basetype AoUnit.
   AggrFunc   aggregate; // The aggregate function.
};

/*
* Basically, joins can only be realized between application
* elements that are linked via a reference defined in the
* model.From the definition of attributes or application
* elements, the references for the joins are determined. It is
* also taken into account that the application elements
* involved are not linked directly. However, there must be an
* unambiguous path between the application elements. The path
* may also include n:m relations. The unambiguousness of
* relations between two application elements no longer exists
* if more than one reference has been defined between the
* application elements. In this case, these references must
* have names and must be indicated explicitly in the
* request.For this purpose, the request structure provides a
* sequence of relation definitions (JoinDefSequence). The
* sequence in which the application elements are addressed in
* the request also determines the sequence in which the
* references between application elements are searched. Thus,
* for every new application element, the server begins with the
* first application element addressed in the request and tries
* to find a relation from there. If no reference to the first
* application element can be found, the search continues with
* the application element that comes next in the request.
* Furthermore, the explicit relation definitions
* (JoinDefSequence) enable an OUTER join, i.e. the result also
* includes those records for which the join could not be
* established.
*/
struct JoinDef {
   T_LONGLONG fromAID;
   T_LONGLONG toAID;
   Name       refName;
   JoinType   joiningType;
};

/*
* The application relation with the instances to create the
* relation with new instances.
*/
struct ApplicationRelationInstanceElementSeq {
   ApplicationRelation     applRel;   // The application relation.
   InstanceElementSequence instances; // The list with instances. The
                                      // application element of the
                                      // instances in the list must match
                                      // one of the application elements
                                      // of the application relation. All
                                      // instances of the list must have
                                      // the same application element.
};

// Sequence definitions (S_xxx).
typedef sequence<T_COMPLEX> S_COMPLEX;
typedef sequence<T_DCOMPLEX> S_DCOMPLEX;
typedef sequence<T_LONGLONG> S_LONGLONG;
typedef sequence<S_LONGLONG> SS_LONGLONG;
typedef sequence<S_COMPLEX> SS_COMPLEX;
typedef sequence<S_DCOMPLEX> SS_DCOMPLEX;
typedef sequence<T_ExternalReference> S_ExternalReference;
typedef sequence<S_ExternalReference> SS_ExternalReference;
typedef sequence<ApplAttr> ApplAttrSequence;
typedef sequence<ApplRel> ApplRelSequence;
typedef sequence<AIDName> AIDNameSequence;
typedef sequence<AIDNameUnitId> AIDNameUnitIdSequence;
typedef sequence<ElemId> ElemIdSequence;
typedef sequence<SelOrder> SelOrderSequence;
typedef sequence<ACL> ACLSequence;
typedef sequence<InitialRight> InitialRightSequence;
typedef sequence<SelAIDNameUnitId> SelAIDNameUnitIdSequence;
typedef sequence<JoinDef> JoinDefSequence;
typedef sequence<ApplicationRelationInstanceElementSeq>
                 ApplicationRelationInstanceElementSeqSequence;


/*
* The application element definition. The same information is
* available at the interface ApplicationElement.
*/
struct ApplElem {
   T_LONGLONG       aid;        // The application element id. The id is
                                // given also with the method getId() at
                                // the interface ApplicationElement. At
                                // the protocol level 3 this information
                                // was stored in the field aiAId of the
                                // structure ApplInfSeq and the request
                                // AOP_GetApplInf.
   Name             beName;     // The base element name, all elements
                                // have a basic element. The same name is
                                // returned by the methods getType() of
                                // the BaseElement interface. The base
                                // element is given with the method
                                // getBaseElement() at the interface
                                // ApplicationElement. At the protocol
                                // level 3 this information was not
                                // delivered  but the corresponding Id of
                                // the base element was stored in the
                                // field aiBId of the structure
                                // ApplInfSeq and the request
                                // AOP_GetApplInf.
   Name             aeName;     // The application element name. The name
                                // is given also with the method
                                // getName() at the interface
                                // ApplicationElement. At the protocol
                                // level 3 this information was stored in
                                // the field aiName of the structure
                                // ApplInfSeq and the request
                                // AOP_GetApplInf.
   ApplAttrSequence attributes; // The attributes of application element.
                                // The atributes are given with the
                                // method getAttributes() of the
                                // interface ApplicationElement. There
                                // are no relations given in this
                                // sequence.
};

/*
* The Union definition for all datatypes.
*/
union TS_Union switch (DataType) {
   case DT_STRING:                        T_STRING
stringVal;
   case DT_SHORT:                         T_SHORT
shortVal;
   case DT_FLOAT:                         T_FLOAT
floatVal;
   case DT_BYTE:                          T_BYTE
byteVal;
   case DT_BOOLEAN:                       T_BOOLEAN
booleanVal;
   case DT_LONG:                          T_LONG
longVal;
   case DT_DOUBLE:                        T_DOUBLE
doubleVal;
   case DT_LONGLONG:                      T_LONGLONG
longlongVal;
   case DT_COMPLEX:                       T_COMPLEX
complexVal;
   case DT_DCOMPLEX:                      T_DCOMPLEX
dcomplexVal;
   case DT_DATE:                          T_DATE
dateVal;
   case DT_BYTESTR:                       T_BYTESTR
bytestrVal;
   case DT_BLOB:                          T_BLOB
blobVal;
   case DS_STRING:                        S_STRING
stringSeq;
   case DS_SHORT:                         S_SHORT
shortSeq;
   case DS_FLOAT:                         S_FLOAT
floatSeq;
```

```
   case DS_BYTE:                         S_BYTE
byteSeq;
   case DS_BOOLEAN:                      S_BOOLEAN
booleanSeq;
   case DS_LONG:                         S_LONG
longSeq;
   case DS_DOUBLE:                       S_DOUBLE
doubleSeq;
   case DS_LONGLONG:                     S_LONGLONG
longlongSeq;
   case DS_COMPLEX:                      S_COMPLEX
complexSeq;
   case DS_DCOMPLEX:                     S_DCOMPLEX
dcomplexSeq;
   case DS_DATE:                         S_DATE
dateSeq;
   case DS_BYTESTR:                      S_BYTESTR
bytestrSeq;
   case DT_EXTERNALREFERENCE:            T_ExternalReference
extRefVal;
   case DS_EXTERNALREFERENCE:            S_ExternalReference
extRefSeq;
   case DT_ENUM:                         T_LONG
enumVal;
   case DS_ENUM:                         S_LONG
enumSeq;
};

/*
* Define a union with sequences of a certain type. Using this
* union instead of sequence <TS_Union> gives much better
* performance.
*/
union TS_UnionSeq switch (DataType) {
   case DT_STRING:                       S_STRING
stringVal;
   case DT_SHORT:                        S_SHORT
shortVal;
   case DT_FLOAT:                        S_FLOAT
floatVal;
   case DT_BYTE:                         S_BYTE
byteVal;
   case DT_BOOLEAN:                      S_BOOLEAN
booleanVal;
   case DT_LONG:                         S_LONG
longVal;
   case DT_DOUBLE:                       S_DOUBLE
doubleVal;
   case DT_LONGLONG:                     S_LONGLONG
longlongVal;
```

```
   case DT_COMPLEX:                     S_COMPLEX
complexVal;
   case DT_DCOMPLEX:                    S_DCOMPLEX
dcomplexVal;
   case DT_DATE:                        S_DATE
dateVal;
   case DT_BYTESTR:                     S_BYTESTR
bytestrVal;
   case DT_BLOB:                        S_BLOB
blobVal;
   case DS_STRING:                      SS_STRING
stringSeq;
   case DS_SHORT:                       SS_SHORT
shortSeq;
   case DS_FLOAT:                       SS_FLOAT
floatSeq;
   case DS_BYTE:                        SS_BYTE
byteSeq;
   case DS_BOOLEAN:                     SS_BOOLEAN
booleanSeq;
   case DS_LONG:                        SS_LONG
longSeq;
   case DS_DOUBLE:                      SS_DOUBLE
doubleSeq;
   case DS_LONGLONG:                    SS_LONGLONG
longlongSeq;
   case DS_COMPLEX:                     SS_COMPLEX
complexSeq;
   case DS_DCOMPLEX:                    SS_DCOMPLEX
dcomplexSeq;
   case DS_DATE:                        SS_DATE
dateSeq;
   case DS_BYTESTR:                     SS_BYTESTR
bytestrSeq;
   case DT_EXTERNALREFERENCE:           S_ExternalReference
extRefVal;
   case DS_EXTERNALREFERENCE:           SS_ExternalReference
extRefSeq;
   case DT_ENUM:                        S_LONG
enumVal;
   case DS_ENUM:                        SS_LONG
enumSeq;
};

// Sequence definitions (S_xxx).
typedef sequence<ApplElem> ApplElemSequence;

/*
* The ASAM ODS value structure.  There is one flag for each
* value. If the union (u) contains a sequence, the flag is
* valid for all values in that sequence.
*
* Meaning of flags:
*    AO_VF_VALID(0x01)      The value is valid.
*    AO_VF_VISIBLE(0x02)    The value has to be
*                           visualized.
*    AO_VF_UNMODIFIED(0x04) The value has not been
*                           modified.
*    AO_VF_DEFINED(0x08)    The value is defined. If
*                           the value in a value matrix
*                           is not available this bit
*                           is not set.
*   The normal value of the flag is 15.
*/
struct TS_Value {
   TS_Union u;    // The value union for values of all known datatypes.
   T_SHORT  flag; // The value flags.
};

/*
* A structure with sequences of a certain type. Using this
* union instead of sequence <TS_Value> gives much better
* performance.
*/
struct TS_ValueSeq {
   TS_UnionSeq u;    // The value union for values of all known
                     // datatypes.
   S_SHORT     flag; // See TS_Value flag.
};

/*
* Application model values. All values of the entire
* application model are stored in this structure and loaded
* to the Client on request. At the protocol level 3 this
* information delivered by the request AOP_GetApplInf and
* AOP_GetAttr for each application element.
*/
struct ApplicationStructureValue {
   ApplElemSequence applElems; // The list of application elements.
   ApplRelSequence  applRels;  // The list of relations in application
                               // model, the relation and the inverse
                               // relation are given.
};

/*
* The ASAM ODS name-value-unit tuple structure with a sequence
* of values.
*/
struct NameValueSeqUnit {
   Name        valName; // Column name or measured quantity name.
   TS_ValueSeq value;   // Column value (vector).
   T_STRING    unit;    // Column unit as string.
};

/*
* AID - Name - Value - UnitId quartet.
*/
struct AIDNameValueUnitId {
   AIDName    attr;   // The attribute  of the application element (aid,
                      // name).
   T_LONGLONG unitId; // The unit of the attribute ot the column. The
                      // unitId is the Id of instance element with the
                      // basetype AoUnit.
   TS_Value   values; // The attribute values with value flags.
};

/*
* AID - Name - Value - UnitId quartet. Multiple values for on
* attribute.
*/
struct AIDNameValueSeqUnitId {
   AIDName     attr;   // The attribute  of the application element (aid,
                       // name).
   T_LONGLONG  unitId; // The unit of the attribute ot the column. The
                       // unitId is the Id of instance element with the
                       // basetype AoUnit.
   TS_ValueSeq values; // The column values with value flags.
};

/*
* The ASAM ODS name-value-unitId tuple structure with a
* sequence of values.
*/
struct NameValueSeqUnitId {
   Name        valName; // Column name or measured quantity name.
   TS_ValueSeq value;   // Column value (vector).
   T_LONGLONG  unitId;  // Column unit as Id.
};

/*
*  Structure for name value query or attribute search
* conditions.
```

```
*/
struct SelValue {
   AIDNameValueUnitId attr;  // The attribute specification with unit of
                             // the value.
   SelOpcode          oper;  // The compare operator between the
                             // attribute and value.
   TS_Value           value; // Value for the condition.
};

/*
* The ASAM ODS name-value-unitid tuple structure. This
* structure is identical with the NameValueUnit, except the
* unit is given as an Id insead of a string.
*/
struct NameValueUnitId {
   Name       valName; // Attribute name or measured quantity name.
   TS_Value   value;   // Attribute value or column value (vector).
   T_LONGLONG unitId;  // Id of attribute or column unit.
};

/*
* The attribute selection structure.
*/
struct SelValueExt {
   AIDNameUnitId attr;  // The attribute specification with unit Id.
   SelOpcode     oper;  // The compare operator between the attribute and
                        // value.
   TS_Value      value; // Value for the condition.
};

/*
* Defines the sequence of selection attributes with their
* logical operators. The Idea is to have the logical operators
* and the selection values in one sequence. Therefore no
* implicit rules are necessary how logical operators have to be
* interpreted to the corresponding selection value.
*/
union SelItem switch (SelType) {
   case SEL_VALUE_TYPE:                 SelValueExt                     value;
   case SEL_OPERATOR_TYPE:              SelOperator
operator;
};

// Sequence definitions (S_xxx).
typedef sequence<NameValueSeqUnit> NameValueSeqUnitSequence;
typedef sequence<AIDNameValueSeqUnitId> AIDNameValueSeqUnitIdSequence;
typedef sequence<SelValue> SelValueSequence;
typedef sequence<NameValueSeqUnitId> NameValueSeqUnitIdSequence;
typedef sequence<SelItem> SelItemSequence;

/*
* The ASAM ODS name-value pair structure.
*/
struct NameValue {
   Name     valName; // Attribute name or measured quantity name.
   TS_Value value;   // Attribute value or column value (vector).
};

/*
* The ASAM ODS name-value-unit tuple structure. This structure
* is identical with the NameValueUnitId, except the unit is
* given as a string insead of an Id.
*/
struct NameValueUnit {
   Name     valName; // Attribute name or measured quantity name.
   TS_Value value;   // Attribute value or column value (vector).
   T_STRING unit;    // Attribute or column unit as string.
};

/*
* The results for one attribute. The result set for all
* attributes are given in the sequence of the result set.
*/
struct AttrResultSet {
   NameValueSeqUnitId      attrValues; // The first 'how_many' results.
                                       // All values have the same AIDName
                                       // and the same UnitId.
   NameValueUnitIdIterator rest;       // The rest of the results.
};

/*
* The query structure.
*
* How to build a query.
*
* A query is a search condition for instances. The instances
* are specified by the values of the attributes. The search
* condition represents an attribute value condition. This means
* the attribute value specifies the selection of the instance
* or instance attribute. An attribute is specified by the
* application element and the name of the attribute (AIDName).
* The conditions are defined in the enumeration SelOPCode. The
* values are given in the TS_Value or TS_Union structure. If we
* like an Unit indepedent condition we needed to define the
* Unit at the attribute, use AIDNameUnitId instead of AIDName,
* the server has to convert the value to the proper attribute
* value. The attribute search condition can be combined by
* operations defined in the enumeration SelOperator. A query
* can be built with a sequence SelValue and SelOperator.
*
* How to read/write the query:
*
* e.g.  SelValue1 AND SelValue2
*
*          selValueSeq    = SelValue1, SelValue2
*          selOPeratorSeq = AND
*
* e.g.  (SelValue1 AND SelValue2) OR SelValue3
*
*          selValueSeq    = SelValue1, SelValue2, SelValue3
*          selOPeratorSeq = OPEN, AND, CLOSE, OR
*
* e.g.  NOT(SelValue1 AND SelValue2) OR SelValue3
*
*          selValueSeq    = SelValue1, SelValue2, SelValue3
*          selOPeratorSeq = NOT, OPEN, AND, CLOSE, OR
*
*  e.g.  NOT(SelValue1) AND SelValue2 OR SelValue3
*
*          selValueSeq    = SelValue1, SelValue2, SelValue3
*          selOPeratorSeq = NOT, AND, OR
*
* There is no selection about the N:M relations.
*
* There are no aggregate functions (MAX, MIN, COUNT etc.)
* defined so we need no "group by" and "having" Clause. All the
* parts defined in the "having"-clause can be defined in the
* select part.
*/
struct QueryStructure {
   AIDNameUnitIdSequence anuSeq;  // The sequence of attributes to be
                                  // reported. At the protocol level 3
                                  // this information was stored in the
                                  // fields applId and nuSeq of the
                                  // structure GetValReq and the request
                                  // AOP_GetVal. At the protocol level 3
                                  // interface only one application
                                  // element could be selected. A pattern
```

```
                                // is accepted for the attribute name.
   SelValueSequence      condSeq; // The query condition sequence. At the
                                // protocol level 3 this information
                                // was stored in the field nsSeq of the
                                // structure GetValReq and the request
                                // AOP_GetVal.
   SelOperatorSequence   operSeq; // The query condition operator
                                // sequence. At the protocol level 3
                                // interface only the operator was
                                // always 'AND'.
   ElemId                relInst; // The related instance. (aid == 0 &&
                                // iid == 0) means no related instance
                                // specified. At the protocol
                                // level 3 this information was stored
                                // in the field elemId of the structure
                                // GetValReq and the request
                                // AOP_GetVal.
   Name                  relName; // Name of the relation. At the
                                // protocol level 3 this information
                                // was stored in the field refName of
                                // the structure GetValReq and the
                                // request AOP_GetVal.
   SelOrderSequence      orderBy; // The order by sequence. The order of
                                // the result set. At the protocol
                                // level 3 interface it was not
                                // possible to set the order.
};

/*
* Result set of one application element.
*/
struct ElemResultSetExt {
   T_LONGLONG                  aid;    // The application element Id.
   NameValueSeqUnitIdSequence values; // The attribute values of the
                                      // instances of the given
                                      // application element.
};

/*
* The extended query structure.
*/
struct QueryStructureExt {
   SelAIDNameUnitIdSequence anuSeq;  // The sequence of attributes to be
                                     // reported. At the protocol level 3
                                     // this information was stored in
                                     // the fields applId and nuSeq of
                                     // the structure GetValReq and the
                                     // request AOP_GetVal. At the
                                     // protocol level 3 interface only
                                     // one application element could be
                                     // selected. A pattern is accepted
                                     // for the attribute name.
   SelItemSequence          condSeq; // The query condition sequence. At
                                     // the protocol level 3 this
                                     // information was stored in the
                                     // field nsSeq of the structure
                                     // GetValReq and the request
                                     // AOP_GetVal.
   JoinDefSequence          joinSeq; // Defined the join between the
                                     // application elements.
   SelOrderSequence         orderBy; // The order by sequence. The order
                                     // of the result set. At the
                                     // protocol level 3 interface it was
                                     // not possible to set the order.
   AIDNameSequence          groupBy; // Defines the grouping attributes
                                     // for a request, necessary if
                                     // aggregate functions are defined
                                     // in the SelAIDNameUnitIdSequence.
};

// Sequence definitions (S_xxx).
typedef sequence<NameValue> NameValueSequence;
typedef sequence<NameValueUnit> NameValueUnitSequence;
typedef sequence<AttrResultSet> AttrResultSetSequence;
typedef sequence<ElemResultSetExt> ElemResultSetExtSequence;

/*
* The result set for one element. The result set for all
* elements are given in the sequence of the result set.
*/
struct ElemResultSet {
   T_LONGLONG            aid;        // The Id of the application
                                     // element.
   AttrResultSetSequence attrValues; // The selected attributes of the
                                     // element. The number of values in
                                     // each AttrResultSet are identical,
                                     // the attributes of one element has
                                     // always the position in the
                                     // AttrResultSet.
};

/*
```

```
* The Result set of the extended query. The iterator is for
* instance oriented access.
*/
struct ResultSetExt {
   ElemResultSetExtSequence    firstElems; // The sequence of the first
                                           // how_many result elements.
   ElemResultSetExtSeqIterator restElems;  // the iterator, which allows
                                           // to iterate above the result
                                           // values, the attributes of
                                           // one instance each
                                           // iteration.
};

// Sequence definitions (S_xxx).
typedef sequence<ElemResultSet> ElemResultSetSequence;
typedef sequence<ResultSetExt> ResultSetExtSequence;

/*
* The ASAM ODS query result types.
*/
interface ResultType {
   const T_SHORT INSTELEM_ITERATOR_AS_RESULT = 0; // Iterator of instance
elements as result of the query (the default).
   const T_SHORT TABLE_ITERATOR_AS_RESULT = 1;    // Iterator for table
access as result type of the query.
   const T_SHORT TABLE_AS_RESULT = 2;             // Table as result type of
the query.
};

/*
* The lock mode of the server. The lock mode tells the way the
* server will lock the objects as soon a modification of the
* server will be done.
*/
interface LockMode {
   const T_SHORT LOCK_INSTANCEELEMENT = 0;    // Lock the instance element.
(Default LockMode)
   const T_SHORT LOCK_APPLICATIONELEMENT = 1; // Lock the application
element, all instances of the application element are locked.
   const T_SHORT LOCK_CHILDREN = 2;           // Lock the children of the
locked object. This mode can be combined with one of the upper two modi.
};

/*
* The bits of the security rights.
*/
interface SecurityRights {
   const T_LONG SEC_READ = 1;   // Read access is allowed.
   const T_LONG SEC_UPDATE = 2; // Update access to an existing object is
allowed.
   const T_LONG SEC_INSERT = 4; // Creating new instances is allowed.
   const T_LONG SEC_DELETE = 8; // Delete of the object is allowed.
   const T_LONG SEC_GRANT = 16;  // Access rights may be passed on.
};

/*
* The security level of an application element.
*/
interface SecurityLevel {
   const T_LONG NO_SECURITY = 0;        // No security defined.
   const T_LONG ELEMENT_SECURITY = 1;   // Security scaled for the
application element.
   const T_LONG INSTANCE_SECURITY = 2;  // Security scaled for instance
elements.
   const T_LONG ATTRIBUTE_SECURITY = 4; // Security scaled for appliation
attributes.
};

/*
* The ASAM ODS query constants.
*/
interface QueryConstants {
   const T_LONG MaxDurationDEFAULT = 0;     // Default value of max duration
parameter of the query  (no limitations).
   const T_STRING MaxDuration = "MaxDuration";            // The ASAM ODS max
duration parameter of the query.
   const T_STRING QueryResultType = "QueryResultType";        // The ASAM ODS
query result type parameter.
   const T_LONG QueryResultTypeDEFAULT =
ResultType::INSTELEM_ITERATOR_AS_RESULT; // Default value of the ASAM ODS
query result type parameter.
};

/*
* The ASAM ODS exception structure.
*/
exception AoException {
   ErrorCode    errCode;
   SeverityFlag sevFlag;
   T_LONG       minorCode;
   T_STRING     reason;
};

/*
```

```
* The ASAM factory interface.
*/
interface AoFactory {

   /* (2001)
   * Get the description of the ASAM ODS factory. If the
   * description is not available an empty string is returned
   * and no exception is thrown. The server loads the
   * description from the base attribute "description" of the
   * instance of AoEnvironment.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *
   * @return  The description of the ASAM ODS factory.
   */
   T_STRING getDescription()
      raises (AoException);

   /* (2002)
   * Get the interface version of the ASAM ODS factory. The
   * interface version is for each ODS version a fixed string.
   * The string for this version is 'OO-5.0'.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *
   * @return  The interface version of the ASAM ODS factory.
   */
   T_STRING getInterfaceVersion()
      raises (AoException);

   /* (2003)
   * Get the name of the ASAM ODS factory. If the name is not
   * available an empty string is returned and no exception is
   * thrown.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *
   * @return  The name of the ASAM ODS factory.
   */
   T_STRING getName()
      raises (AoException);

   /* (2004)
   * Get the type of the ASAM ODS factory. If the type is not
   * available an empty string is returned and no exception is
   * thrown. The server loads the type from the base attribute
   * "Application_model_type" of the instance of AoEnvironment.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *
   * @return  The type of the ASAM ODS factory.
   */
   T_STRING getType()
      raises (AoException);

   /* (2005)
   * Establish a new session to an ASAM ODS server. The server
   * normally checks the activity of the session and will close
   * the session after a time period of inactivity.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECT_FAILED
   *    AO_CONNECT_REFUSED
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_OPEN_MODE_NOT_SUPPORTED
   *    AO_SESSION_LIMIT_REACHED
   *
   * @param  auth  A string that may contain authentication
   *               information. The following values are
   *               currently supported:
   *                  USER
   *                  PASSWORD
```

```
   *                  OPENMODE
   *               The values may be specified in any order and
   *               have to be separated by comma.
   *
   *               Example:
   *               "USER=hans, PASSWORD=secret, OPENMODE=read"
   *
   * @return  The new created ASAM ODS session.
   */
   AoSession newSession(
      in T_STRING auth)
      raises (AoException);

}; // Interface AoFactory.

/*
* The ASAM ODS session interface.
*/
interface AoSession {

   /* (3001)
   * Abort (rollback) a transaction. The changes made in the
   * transaction are lost.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_ACCESS_DENIED
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *    AO_TRANSACTION_NOT_ACTIVE
   */
   void abortTransaction()
      raises (AoException);

   /* (3002)
   * Close session to an ASAM ODS server. Active transactions
   * are committed.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   */
   void close()
      raises (AoException);

   /* (3003)
   * Commit a transaction. The changes made in the transaction
   * become permanent.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_ACCESS_DENIED
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *    AO_TRANSACTION_NOT_ACTIVE
   */
   void commitTransaction()
      raises (AoException);

   /* (3004)
   * Get the application model from the current session.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_ACCESS_DENIED
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The application model.
   */
   ApplicationStructure getApplicationStructure()
      raises (AoException);

   /* (3005)
   * Get the application model as values from the current
   * session.
   *
   * @throws AoException
   * with the following possible error codes:
```

```
*    AO_ACCESS_DENIED
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The application model as value.
*/
ApplicationStructureValue getApplicationStructureValue()
   raises (AoException);

/* (3006)
* Get the ASAM ODS base model from the current session. The
* complete base model is returned. This base model has all
* possible base elements with all possible base attributes.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The base model.
*/
BaseStructure getBaseStructure()
   raises (AoException);

/* (3007)
* Get context variables from the session. A pattern string
* can be specified to select groups of variables.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_FOUND
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  varPattern  The name or the search pattern for the
*                     context variable(s).
```

```
*
* @return  A list of context variables.
*/
NameValueIterator getContext(
   in Pattern varPattern)
   raises (AoException);

/* (3008)
* Get a context variable by its name from the session.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_FOUND
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  varName  The name of the requested context
*                  variable.
*
* @return  The requested context variable.
*/
NameValue getContextByName(
   in Name varName)
   raises (AoException);

/* (3009)
* List the names of context variables from the session. A
* pattern string can be specified to select groups of
* variables.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NOT_FOUND
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  varPattern  The name or the search pattern for the
*                     context variable(s).
```

```
*
* @return  A list of context variable names.
*/
NameIterator listContext(
   in Pattern varPattern)
   raises (AoException);

/* (3010)
* Remove context variables from the session. A pattern
* string can be specified to remove groups of variables.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NOT_FOUND
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  varPattern  The name or the search pattern for the
*                     context variable(s) to be removed.
*/
void removeContext(
   in Pattern varPattern)
   raises (AoException);

/* (3011)
* Set/modify a known context variable or add a new context
* variable to the session.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  contextVariable  The context variable.
*/
void setContext(
   in NameValue contextVariable)
   raises (AoException);
```

```
/* (3012)
* Set/modify a known context variable or add a new context
* variable to the session. This is a convienience method for
* the frequently used string variable type. It uses
* setContext  internally.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  varName  The name of the context variable.
*
* @param  value  The value of the context variable.
*/
void setContextString(
   in Name varName,
   in T_STRING value)
   raises (AoException);

/* (3013)
* Start a transaction on the physical storage system (e.g.
* database system). Only when a transaction is started it is
* allowed to create or modify instances or measurement data.
* The changes get permanent with a commit of the transaction
* or will be lost with an abort of the transaction. If the
* session is closed the transaction will be committed
* automatically. If a transaction is already active an
* exception is thrown.
*
* @throws AoException
* with the following possible error codes:
*    AO_ACCESS_DENIED
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_ALREADY_ACTIVE
*/
void startTransaction()
```

```
   raises (AoException);

/* (3014)
* Make the changes permanent.
*
* @throws AoException
* with the following possible error codes:
*    AO_ACCESS_DENIED
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*/
void flush()
   raises (AoException);

/* (3015)
* Every new created instance will set its initial rights to
* <acl> . This method overrides the default-methods for
* applying initial rights. The initial rights are only valid
* for the current session.
*
* @throws AoException
* with the following possible error codes:
*    AO_ACCESS_DENIED
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  irlEntries  The current initial rights.
*
* @param  set  Set (1) or remove (0) the current initial
*              rights. The previous initial rights get lost.
*              If the parameter set is 0 (remove) the
*              parameter irlEntries will be ignored.
*/
void setCurrentInitialRights(
   in InitialRightSequence irlEntries,
   in T_BOOLEAN set)
   raises (AoException);
```

```
/* (3016)
* Get the current lock mode. The lock mode tells the server
* which objects to lock for upcoming changes. Application
* elements, instance elements or children of elements can be
* locked.
*
* @throws AoException
* with the following possible error codes:
*    AO_ACCESS_DENIED
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @return  The current lock mode. The lock mode constants
*          are defined in the interface LockMode. The
*          interface definition language IDL does not allow
*          to set the values of enumerations thus the
*          constant definitions had to be done in an
*          interface.
*/
T_SHORT getLockMode()
   raises (AoException);

/* (3017)
* Set the new lock mode. The lock mode tells the server
* which objects to lock for upcoming changes. Application
* elements, instance elements or children of elements can be
* locked.
*
* @throws AoException
* with the following possible error codes:
*    AO_ACCESS_DENIED
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  lockMode  The new lock mode. The lock mode
*                    constants are defined in the interface
*                    LockMode. The interface definition
*                    language IDL does not allow to set the
*                    values of enumerations thus the constant
*                    definitions had to be done in an
*                    interface.
*/
void setLockMode(
   in T_SHORT lockMode)
   raises (AoException);

/* (3018)
* Get the application element access object from the current
* session.
*
* @throws AoException
* with the following possible error codes:
*    AO_ACCESS_DENIED
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The application element access object.
*/
ApplElemAccess getApplElemAccess()
   raises (AoException);

/* (3019)
* Change the password for user defined by <username> to
* <newPassword>. A normal user must supply his current
* password <oldPassword>. The superuser can change the
* password without supplying the current password
* <oldPassword>. If no username is given the password of the
* user of te current session will be changed. The password
* is normally encrypted in the attribute of the user
* instance element. Creating a new user can be done by
* creating a new instance, afterwards the password must be
* set by the superuser.
*
* @throws AoException
* with the following possible error codes:
*    AO_ACCESS_DENIED
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*    AO_WRONG_PASSWORD
*
* @param  username  The name of the user for which the
*                   password will be changed. If no username
*                   is given the password of the current
*                   user will be changed. If the username
*                   differs from the current user the
*                   current user must be a superuser.
*
* @param  oldPassword  The current password of the user. A
*                      normal user must supply his current
*                      password. The superuser can change
*                      the password without supplying the
*                      current password.
*
* @param  newPassword  The new password of the user.
*/
void setPassword(
   in T_STRING username,
   in T_STRING oldPassword,
   in T_STRING newPassword)
   raises (AoException);

/* (3020)
* Get the description of the ASAM ODS session.The
* description of the session is identical with description
* of the ASAM ODS factory. If the description is not
* available an empty string is returned and no exception is
* thrown. The server loads the description from the base
* attribute "description" of the instance of AoEnvironment.
*
* @throws AoException
* with the following possible error codes:
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_CONNECTION_LOST
*
* @return  The description of the ASAM ODS session.
*/
```

```
T_STRING getDescription()
   raises (AoException);

/* (3021)
* Get the name of the ASAM ODS session. The name of the
* session is identical with the name of the ASAM ODS
* factory. If the name is not available an empty string is
* returned and no exception is thrown. The server loads the
* description from the base attribute "name" of the instance
* of AoEnvironment.
*
* @throws AoException
* with the following possible error codes:
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_CONNECTION_LOST
*
* @return  The name of the ASAM ODS session.
*/
Name getName()
   raises (AoException);

/* (3022)
* Get the type of the ASAM ODS session. The type of the
* session is identical with the type of the ASAM ODS
* factory. If the type is not available an empty string is
* returned and no exception is thrown. The server loads the
* type from the base attribute "Application_model_type" of
* the instance of AoEnvironment.
*
* @throws AoException
* with the following possible error codes:
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_CONNECTION_LOST
*
* @return  The type of the ASAM ODS session.
*/
T_STRING getType()
   raises (AoException);

/* (3023)
```

```
   * Create a QueryEvaluator object.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_ACCESS_DENIED
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The new created query evaluator object
   */
   QueryEvaluator createQueryEvaluator()
      raises (AoException);

   /* (3024)
   * Create a new object with the Interface Blob on the server.
   * This object can be used to create an attribute value of
   * the datatype DT_BLOB.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The reference of the blob object which is
   *          generated at the server.
   */
   Blob createBlob()
      raises (AoException);

}; // Interface AoSession.

/*
* The ASAM ODS application attribute interface.
*/
interface ApplicationAttribute {

   /* (4001)
   * Get the base attribute of the application attribute.
```

```
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The base attribute of the application attribute.
*          A 'null' is returned if the application attribute
*          has no base attribute.
*/
BaseAttribute getBaseAttribute()
   raises (AoException);

/* (4002)
* Get the data type of the application attribute.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The data type of the application attribute.
*/
DataType getDataType()
   raises (AoException);

/* (4003)
* Get the maximum allowed length of the value of the
* application attribute.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The maximum allowed length of the application
*          attribute.
```

```
*/
T_LONG getLength()
   raises (AoException);

/* (4004)
* Get the name of the application attribute.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The name of the application attribute.
*/
Name getName()
   raises (AoException);

/* (4005)
* Get the unit Id of the application attribute. The unit Id
* is only valid for the current server.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The unit Id of the application attribute.
*/
T_LONGLONG getUnit()
   raises (AoException);

/* (4006)
* Get the obligatory flag of the application attribute.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The obligatory flag of the application attribute.
*/
T_BOOLEAN isObligatory()
   raises (AoException);

/* (4007)
* Get the unique flag of the application attribute.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The unique flag of the application attribute.
*/
T_BOOLEAN isUnique()
   raises (AoException);

/* (4008)
* Set the base attribute of the application attribute. This
* allows the client to declare the application attribute
* (new or existing) additional to a base attribute. The
* application attribute will become the derived attribute of
* the given base attribute.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The base attribute must be unique within the application
* element otherwise the exception
* AO_DUPLICATE_BASE_ATTRIBUTE is thrown.
*
* For performance and flexibility reasons this set-method
* should be used before the new application attribute is
* committed the first time.
*
* If this method is called before the first commit it will
* not throw the following exceptions:
*    AO_INVALID_DATATYPE
*    AO_MISSING_VALUE
*    AO_NOT_UNIQUE.
```

```
*
* After the first commit, there may be instances of the
* application attribute. These instances may cause the
* following problems:
*
* AO_INVALID_DATATYPE: The datatype of the base attribute is
* not the same as the datatype of the instanciated
* attributes.
*
* AO_MISSING_VALUE: The obligatory flag of the base
* attribute is set but there are one or more empty values in
* the instances.
*
* AO_NOT_UNIQUE: The unique flag of the base attribute is
* set but the values of the instances are not unique.
*
* The length, the name and the unit of the application
* attribute are not affected by this call.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_DUPLICATE_BASE_ATTRIBUTE
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_DATATYPE
*    AO_MISSING_VALUE
*    AO_NOT_IMPLEMENTED
*    AO_NOT_UNIQUE
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  baseAttr  The base attribute.
*/
void setBaseAttribute(
   in BaseAttribute baseAttr)
   raises (AoException);

/* (4009)
* Set the data type of the application attribute.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* It is not allowed to set the datatype of application
* attributes that represent base attributes. An attempt to
* set the datatype of such an application attribute will
* result in the exception AO_IS_BASE_ATTRIBUTE.
*
* For performance and flexibility reasons this set-method
* should be used before the new application attribute is
* committed the first time.
*
* If this method is called before the first commit it will
* not throw the following exception:
*    AO_INVALID_DATATYPE
*
* After the first commit, there may be instances of the
* application attribute. These instances may cause the
* following problem:
*
* AO_INVALID_DATATYPE: The datatype of the base attribute is
* not the same as the datatype of the instanciated
* attributes.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_HAS_BASE_ATTRIBUTE
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_DATATYPE
*    AO_IS_BASE_ATTRIBUTE
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aaDataType  The data type.
*/
void setDataType(
   in DataType aaDataType)
   raises (AoException);

/* (4010)
* Set the obligatory flag of the application attribute.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* It is not allowed to set the obligatory flag of
* application attributes
* that represent base attributes. An attempt to set the
* obligatory flag of such an application attribute will
* result in the exception AO_IS_BASE_ATTRIBUTE.
*
* For performance and flexibility reasons this set-method
* should be used before the new application attribute is
* committed the first time.
*
* If this method is called before the first commit it will
* not throw the following exception:
*    AO_MISSING_VALUE
*
* After the first commit, there may be instances of the
* application attribute. These instances may cause the
* following problem:
*
* AO_MISSING_VALUE: The obligatory flag of the base
* attribute is set but there are one or more empty values in
* the instances.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_HAS_BASE_ATTRIBUTE
*    AO_IMPLEMENTATION_PROBLEM
*    AO_IS_BASE_ATTRIBUTE
*    AO_MISSING_VALUE
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aaIsObligatory  The obligatory flag.
*/
void setIsObligatory(
   in T_BOOLEAN aaIsObligatory)
   raises (AoException);

/* (4011)
* Set the unique flag of the application attribute.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The server will check if the values of the instance
* attributes are unique. If this flag is set and the values
* of an attribute are not unique when using the method
* setValue an exception is thrown. If instances of the
* application element already exist that contain non-unique
* values and the flag shall be set this method throws an
* exception.
*
* It is not allowed to set the unique flag of application
* attributes that represent base attributes. An attempt to
* set the unique flag of such an application attribute will
* result in the exception AO_IS_BASE_ATTRIBUTE.
*
* If the unique flag is set to TRUE the obligatory flag is
* also set to TRUE. The previous values of both flag do not
* matter in this case. Setting the unique flag to FALSE does
* not affect the obligatory flag.
*
* For performance and flexibility reasons this set-method
* should be used before the new application attribute is
* committed the first time.
*
* If this method is called before the first commit it will
* not throw the following exception:
*    AO_MISSING_VALUE
*    AO_NOT_UNIQUE
*
* After the first commit, there may be instances of the
* application attribute. These instances may cause the
* following problem:
*
* AO_MISSING_VALUE: The obligatory flag of the base
* attribute is set but there are one or more empty values in
* the instances.
*
* AO_NOT_UNIQUE: The unique flag of the base attribute is
* set but the values of the instances are not unique.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_HAS_BASE_ATTRIBUTE
*    AO_IMPLEMENTATION_PROBLEM
*    AO_IS_BASE_ATTRIBUTE
*    AO_MISSING_VALUE
*    AO_NOT_IMPLEMENTED
*    AO_NOT_UNIQUE
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aaIsUnique  The unique flag.
*/
void setIsUnique(
   in T_BOOLEAN aaIsUnique)
   raises (AoException);

/* (4012)
* Set the maximum allowed length of the application
* attribute.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* This method is useful for ODS database design tools.
* Negative length values are not allowed.
*
* This method provides only a hint to a database server in
* the design phase which size the data entries may have. The
* length is ignored for all other datatypes than DT_STRING
* and DS_*.
*
* For performance and flexibility reasons this set-method
* should be used before the new application attribute is
* committed the first time.
*
* If this method is called before the first commit it will
* not throw the following exception:
*    AO_HAS_INSTANCES
*
* After the first commit, there may be instances of the
* application attribute. These instances may cause the
* exception AO_HAS_INSTANCES if the instances of the
* application attribute are not empty.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_HAS_INSTANCES
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_LENGTH
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
```

```
*
* @param  aaLength  The maximum attribute length.
*/
void setLength(
   in T_LONG aaLength)
   raises (AoException);

/* (4013)
* Set the name of an application attribute.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The name must be unique.
*
* For performance and flexibility reasons this set-method
* should be used before the new application attribute is
* committed the first time.
*
* The name of an application attribute must not exceed the
* maximum name length of the underlying physical storage.
* The current specification of the physical storage restricts it to 30
* characters.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_DUPLICATE_NAME
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aaName  The application attribute name.
*/
void setName(
   in Name aaName)
   raises (AoException);

/* (4014)
* Set the unit Id of an application attribute.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
```

```
* The unit Id is only valid for the current server. If
* instances of the application attribute exist, the
* respective values are automatically converted to the new
* unit. If there is no known conversion an exception is
* thrown.
*
* The automatic conversion can be avoided if the unit is set
* to zero. After that the transaction must be committed. In
* the next step the new unit may be set in another
* transaction.
*
* The automatic conversion is done only for the following
* datatypes:
*    DT_BYTE
*    DT_COMPLEX
*    DT_DCOMPLEX
*    DT_DOUBLE
*    DT_FLOAT
*    DT_LONG
*    DT_LONGLONG
*    DT_SHORT
* as well as for the corresponding sequence datatypes. For
* complex datatypes the real and imaginary part are
* converted separately.
*
* If the unit of an attribute is set the unit is constant.
* If the value of the attribute has another unit the value
* is calibrated to the unit of the application attribute. If
* there is no unit at the application attribute the unit at
* the attribute value is stored and reported on request at
* the instance.
*
* For performance and flexibility reasons this set-method
* should be used before the new application attribute is
* committed the first time.
*
* If this method is called before the first commit it will
* not throw the following exceptions:
*    AO_INCOMPATIBLE_UNITS
*    AO_MATH_ERROR
*
* After the first commit, there may be instances of the
* application attribute. These instances may cause the
* following problems:
*
* AO_INCOMPATIBLE_UNITS: No conversion rules is known to
* convert the unit.
*
* AO_MATH_ERROR: Converting the values to the new unit
* results in data overflow or underflow or a division by
* zero is detected.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INCOMPATIBLE_UNITS
*    AO_MATH_ERROR
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_UNKNOWN_UNIT
*
* @param  aaUnit  The unit Id.
*/
void setUnit(
   in T_LONGLONG aaUnit)
   raises (AoException);

/* (4015)
* The given usergroup the rights should be set for. <rights>
* defines the rights to set or to clear. If the parameter
* <set> is set to 'set', the rights in <rights> are set,
* all others are cleared. If the parameter <set> is set to
* 'add', the rights in <rights> are added to the existing
* rights. If the parameter <set> is set to 'remove', the
* rights in <rights> are removed from the existing rights.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  usergroup  The usergroup for which the rights will
*                    be modified.
*
```

```
* @param  rights  The new right for the usergroup. The
*                 rights constants are defined in the
*                 interface SecurityRights. The interface
*                 definition language IDL does not allow to
*                 set the values of enumerations thus the
*                 constant definitions had to be done in an
*                 interface.
*
* @param  set  What to do with the new right.
*/
void setRights(
   in InstanceElement usergroup,
   in T_LONG rights,
   in RightsSet set)
   raises (AoException);

/* (4016)
* Retrieve access control list information of the given
* object.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @return  The access control list entries of the given
*          application element.
*/
ACLSequence getRights()
   raises (AoException);

/* (4017)
* Return the application element to which the attribute
* belongs.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The application element of the attribute.
*/
ApplicationElement getApplicationElement()
   raises (AoException);

/* (4018)
* Get the autogenerate flag of the application attribute.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The autogenerate flag of the application
*          attribute.
*/
T_BOOLEAN isAutogenerated()
   raises (AoException);

/* (4019)
* Set the autogenerate flag of the application attribute.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* For performance and flexibility reasons this set-method
* should be used before the new application attribute is
* committed the first time.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_HAS_BASE_ATTRIBUTE
*    AO_IMPLEMENTATION_PROBLEM
*    AO_IS_BASE_ATTRIBUTE
*    AO_MISSING_VALUE
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
```

```
*
* @param  isAutogenerated  The autogenerate flag.
*/
void setIsAutogenerated(
   in T_BOOLEAN isAutogenerated)
   raises (AoException);

/* (4020)
* Get the definition of the enumeration.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @return  The ASAM ODS enumeration.
*/
EnumerationDefinition getEnumerationDefinition()
   raises (AoException);

/* (4021)
* Set the definition of the enumeration. This method
* modifies the application model, only the superuser can use
* this method.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*    AO_ACCESS_DENIED
*
* @param  enumDef  The new enumeration definition.
*/
void setEnumerationDefinition(
   in EnumerationDefinition enumDef)
   raises (AoException);
```

```
/* (4022)
* Has the attribute an unit. If this flag is set, all the
* attributes of the instances derived from this attribute
* will have an unit.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @return  The flag if the attribute has an unit.
*/
T_BOOLEAN hasUnit()
   raises (AoException);

/* (4023)
* Set whether the attribute will have a unit or not. A call to
* the method setUnit() will automatically set the
* withUnit(TRUE).
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  withUnit  The flag; TRUE if the attribute will have a
*                   unit.
*/
void withUnit(
   in T_BOOLEAN withUnit)
   raises (AoException);

/* (4024)
* Has the attribute a value flag. If this flag is set, all
* the attributes of the instances derived from this
```

```
   * attribute will have a value flag. If this flag is not set
   * the flag in the TS_Value structure can be ignored.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *    AO_TRANSACTION_NOT_ACTIVE
   *
   * @return  The flag if the attribute has a value flag.
   */
   T_BOOLEAN hasValueFlag()
      raises (AoException);

   /* (4025)
   * Specifies whether the attribute will have a value flag or not. If
   * this flag isn't set the flag of the TS_Value will be
   * ignored by the server.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *    AO_TRANSACTION_NOT_ACTIVE
   *
   * @param  withValueFlag  The flag; TRUE if the attribute will have a
   *                        value flag.
   */
   void withValueFlag(
      in T_BOOLEAN withValueFlag)
      raises (AoException);

}; // Interface ApplicationAttribute.

/*
* The ASAM ODS application element interface.
*/
interface ApplicationElement {
```

```
/* (5001)
* Create a new application attribute on the server.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The properties of the new application attribute may be
* changed via the set-methods of the ApplicationAttribute
* interface.
*
* For performance reasons it is recommended to set all
* required properties of an application attribute before it
* is committed the first time. This avoids database
* cross-checks for each attribute.
*
* The default properties of a new application attribute
* are:
*    BaseAttribute   NULL
*    DataType        DT_UNKNOWN
*    IsObligatory    0
*    IsUnique        0
*    Length          0
*    Name            "AUTOGEN"
*    Unit            NULL
*
* If there are already instances of the application element
* the values of the existing instances of the new attribute
* are set to undefined (flag AO_VF_DEFINED is set to
* zero).
*
* The exception AO_DUPLICATE_NAME name occurs if there is
* already another application attribute with the name
* "AUTOGEN".
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_DUPLICATE_NAME
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The new application attribute.
*/
```

```
ApplicationAttribute createAttribute()
   raises (AoException);

/* (5002)
* Create an instance of the application element.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The instance gets permanent when the  transaction is
* committed.  All attributes connected to the application
* element are automatically created and connected to the
* instance. The values of the attributes can be set by the
* method setValue of the interface InstanceElement.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  ieName  The instance name.
*
* @return  The new instance.
*/
InstanceElement createInstance(
   in Name ieName)
   raises (AoException);

/* (5003)
* Get a list of all related application elements connected
* to this application element.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The related application elements.
*/
```

```
ApplicationElementSequence getAllRelatedElements()
   raises (AoException);

/* (5004)
* Get a list of all application relations connected to this
* application element. The inverse relation of relations
* connected to other application elements pointing to the
* given application elements are not returned.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The application relations of the application
*          element.
*/
ApplicationRelationSequence getAllRelations()
   raises (AoException);

/* (5005)
* Get the application attribute of an application element
* which is inherited from the base attribute with the given
* name. The base name is case insensitive and may not
* contain wildcard characters.
*
* Note: The base model is case blind,  e.g. Id, ID and id is
* all the same base attribute.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  baName  The base attribute name.
*
* @return  The application attribute.
*/
ApplicationAttribute getAttributeByBaseName(
   in Name baName)
   raises (AoException);

/* (5006)
* Get the application attribute of an application element
* which has the given name. The name is case sensitive and
* may not contain wildcard characters.
*
* Note: The application model is case sensitive, eg Id and
* ID are different application attributes, don't use this
* misleading attribute name.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aaName  The application attribute name.
*
* @return  The application attribute.
*/
ApplicationAttribute getAttributeByName(
   in Name aaName)
   raises (AoException);

/* (5007)
* Get a list of the application attributes of an application
* element. The reference attributes are not returned.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aaPattern  The name or the search pattern for the
*                    application attribute name.
*
* @return  The application attributes.
```

```
*/
ApplicationAttributeSequence getAttributes(
   in Pattern aaPattern)
   raises (AoException);

/* (5008)
* Get the base element of an application element.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The base element.
*/
BaseElement getBaseElement()
   raises (AoException);

/* (5009)
* Get the Id of an application element.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The Id of the application element.
*/
T_LONGLONG getId()
   raises (AoException);

/* (5010)
* Get the instance element specified by the given Id. If the
* Id of the instance is not unique an exception is thrown.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  ieId  The instance element Id.
*
* @return  The instance element.
*/
InstanceElement getInstanceById(
   in T_LONGLONG ieId)
   raises (AoException);

/* (5011)
* Get the instance element specified by the given name. If
* the name of the instance is not unique an exception is
* thrown.
*
* This is a convienience method for instance elements with
* unique names. If there are duplicate names for instance
* use the method getInstances instead and specify the
* requested name as pattern parameter.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_DUPLICATE_NAME
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  ieName  The instance element name.
*
* @return  The instance element.
*/
InstanceElement getInstanceByName(
   in Name ieName)
   raises (AoException);

/* (5012)
* Get the instances whose names match the pattern. The
* pattern is case sensitive and may contain wildcard
* characters.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  iePattern  The name or the search pattern for the
*                    instance element name.
*
* @return  The instance elements.
*/
InstanceElementIterator getInstances(
   in Pattern iePattern)
   raises (AoException);

/* (5013)
* Get the name of an application element.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The name of the application element.
*/
Name getName()
   raises (AoException);

/* (5014)
* Get related application elements connected via the
* specified relationship.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_RELATIONSHIP
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
```

```
*
* @param  aeRelationship  The requested relationship.
*
* @return  The related application elements.
*/
ApplicationElementSequence getRelatedElementsByRelationship(
   in Relationship aeRelationship)
   raises (AoException);

/* (5015)
* Get application relations of the requested type connected
* from this application element. The inverse relations are
* not returned.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_RELATION_TYPE
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aeRelationType  The requested relation type.
*
* @return  The application relations.
*/
ApplicationRelationSequence getRelationsByType(
   in RelationType aeRelationType)
   raises (AoException);

/* (5016)
* Get the names of all related application elements.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The names of the related application elements.
*/
NameSequence listAllRelatedElements()
```

```
   raises (AoException);

/* (5017)
* Get the application attribute names of the application
* element. There are no attribute names returned in the
* result list that contain a reference to another
* application element.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aaPattern  The name or the search pattern for the
*                    application attribute name.
*
* @return  The names of the application attributes.
*/
NameSequence listAttributes(
   in Pattern aaPattern)
   raises (AoException);

/* (5018)
* Get the names of the instances whose names match the
* pattern. The pattern is case sensitive and may contain
* wildcard characters.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aaPattern  The name or the search pattern for the
*                    application attribute name.
*
* @return  The names of the instances.
*/
NameIterator listInstances(
   in Pattern aaPattern)
   raises (AoException);

/* (5019)
* Get the names of related application elements connected
* via the specified relationship.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_RELATIONSHIP
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aeRelationship  The requested relationship.
*
* @return  The names of the related application elements.
*/
NameSequence listRelatedElementsByRelationship(
   in Relationship aeRelationship)
   raises (AoException);

/* (5020)
* Remove an application attribute from an application
* element. If there are instances of the application element
* the attribute of the existing instances change from
* application to instance attributes.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_FOUND
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  applAttr  The application attribute to remove.
*/
```

```
void removeAttribute(
   in ApplicationAttribute applAttr)
   raises (AoException);

/* (5021)
* Remove an instance from the application element.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The instance is removed from the server when the
* transaction is committed. If the recursive flag is set all
* children of the instance are also deleted. Removing
* instances is allowed only if there are no
* references(relations) to this instance. If the recursive
* flag is set a reference to one of the children is not
* allowed and will cause an exception.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_HAS_REFERENCES
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_FOUND
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  ieId  The instance Id.
*
* @param  recursive  The recursive flag.
*/
void removeInstance(
   in T_LONGLONG ieId,
   in T_BOOLEAN recursive)
   raises (AoException);

/* (5022)
* Set the base element of the application element.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The assignment to the current base element is overwritten.
* If there are instances of the application element or
* references to the application element an exception is
* thrown.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_HAS_INSTANCES
*    AO_HAS_REFERENCES
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  baseElem  The base element.
*/
void setBaseElement(
   in BaseElement baseElem)
   raises (AoException);

/* (5023)
* Set the name of the application element.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The name of the application element must be unique.
*
* The name of an application element must not exceed the
* maximum name length of the underlying physical storage.
* The current physical storage specification restricts it to 30
* characters.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_DUPLICATE_NAME
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aeName  The application element name.
*/
void setName(
   in Name aeName)
   raises (AoException);

/* (5024)
* The given usergroup the rights should be set for. <rights>
* defines the rights to set or to clear. If the parameter
* <set> is set to 'set', the rights in <rights> are set, all
* others are cleared. If the parameter <set> is set to
* 'add', the rights in <rights> are added to the existing
* rights. If the parameter <set> is set to 'remove', the
* rights in <rights> are removed from the existing rights.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  usergroup  The usergroup for which the rights will
*                    be modified.
*
* @param  rights  The new right for the usergroup. The
*                 rights constants are defined in the
*                 interface SecurityRights. The interface
*                 definition language IDL does not allow to
*                 set the values of enumerations thus the
*                 constant definitions had to be done in an
*                 interface.
*
* @param  set  What to do with the new right.
*/
void setRights(
   in InstanceElement usergroup,
   in T_LONG rights,
   in RightsSet set)
   raises (AoException);

/* (5025)
* Retrieve access control list information of the given
* object.
*
* @throws AoException
```

```
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @return  The access control list entries of the given
*          application element.
*/
ACLSequence getRights()
   raises (AoException);

/* (5026)
* Retrieve access control list information for the initial
* rights of the given object.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @return  The access control list entries with the initial
*          rights of the given application element.
*/
InitialRightSequence getInitialRights()
   raises (AoException);

/* (5027)
* The given usergroup the initial rights should be set for.
* <rights> defines the rights to set or to clear. If the
* parameter <set> is set to 'set', the rights in <rights>
* are set, all others are cleared. If the parameter <set> is
* set to 'add', the rights in <rights> are added to the
* existing rights. If the parameter <set> is set to
* 'remove', the rights in <rights> are removed from the
* existing rights.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  usergroup  The usergroup for which the initial
*                    rights will be modified.
*
* @param  rights  The new initial rights for the
*                usergroup.The rights constants are defined
*                in the interface SecurityRights. The
*                interface definition language IDL does not
*                allow to set the values of enumerations
*                thus the constant definitions had to be
*                done in an interface.
*
* @param  refAid  The Id of referencing application element
*                for which the initial rights will be used.
*                If no refAid is set the initial rights
*                will be used for each new instance element
*                independent of the application element.
*
* @param  set  What to do with the new initial rights.
*/
void setInitialRights(
   in InstanceElement usergroup,
   in T_LONG rights,
   in T_LONGLONG refAid,
   in RightsSet set)
   raises (AoException);

/* (5028)
* Set for the given application element, which relation will
* be used to determine the initial rights for the new
* created instances.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  applRel  The application relation which will be
*                  used to determine the initial rights. The
*                  relation range of the application
*                  relation must be [1:1] otherwise the
*                  server can not find a unique instance
*                  element to retrieve the initial rights.
*
* @param  set  Set or remove the relation for the initial
*              rights. If this parameter is true the
*              relation will be set otherwise removed.
*/
void setInitialRightRelation(
   in ApplicationRelation applRel,
   in T_BOOLEAN set)
   raises (AoException);

/* (5029)
* Get all relations which are used to retrieve the instances
* to create the initial rights of the new created instance
* element. If there are more then one application relation
* the initial rights of each related instance are 'ored' to
* the list of the initial rights.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @return  The sequence with the application relations which
*          will be used to create the initial rights of the
*          new created instance element.
*/
ApplicationRelationSequence getInitialRightRelations()
   raises (AoException);

/* (5030)
* Get the security level of the application element. The
* security level tells if there is a security check for both
* application element and instance elements or only for the
* application attributes, the instance elements or none at
* all.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @return  The current security level. The security level
*          constants are defined in the interface
*          SecurityLevel. The interface definition language
*          IDL does not allow to set the values of
*          enumerations thus the constant definitions had to
*          be done in an interface.
*/
T_LONG getSecurityLevel()
   raises (AoException);

/* (5031)
* Set the security level for the given application element.
* If the security level is added the client is responsable
* for the access control list entries of the existing
* objects.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  secLevel  The new security level.The security
*                   level constants are defined in the
*                   interface SecurityLevel. The interface
*                   definition language IDL does not allow
*                   to set the values of enumerations thus
*                    the constant definitions had to be done
*                    in an interface.
*
* @param  set  What to do with the new security level.
*/
void setSecurityLevel(
   in T_LONG secLevel,
   in RightsSet set)
   raises (AoException);

/* (5032)
* Get the application model to which the application
* element belongs. The same application structure will be
* returned as the object from the method
* getApplicationStructure of the Interface AoSession. This
* method guarantees that the client software is  able to
* return to the session without the session object is
* available.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @return  The application model to which the
*          application element belongs.
*/
ApplicationStructure getApplicationStructure()
   raises (AoException);

/* (5033)
* Create a list with instances. The attribute are given with
* the name of the sequence. The values of the attributes are
* given in the value sequence. The index in the different
* value sequences match for one instance element. The index
* in the instance element sequence of the related instances
* match for the instance with the same index in the value
* sequence.
*
* @throws AoException
* with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *    AO_TRANSACTION_NOT_ACTIVE
   *    AO_INVALID_REQUEST
   *
   * @param  attributes  The attributes of the new created
   *                     instances.
   *
   * @param  relatedInstances  The list with related instances
   *                           for different application
   *                           relations.
   *
   * @return  The list with the new created instances.
   */
   InstanceElementSequence createInstances(
      in NameValueSeqUnitSequence attributes,
      in ApplicationRelationInstanceElementSeqSequence relatedInstances)
      raises (AoException);

}; // Interface ApplicationElement.

/*
* The ASAM ODS application relation interface.
*/
interface ApplicationRelation {

   /* (6001)
   * Get the base relation of the application relation.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The base relation of the application relation. A
   *          'null' is returned if the application relation
   *          has no base relation.
   */
   BaseRelation getBaseRelation()
      raises (AoException);

   /* (6002)
   * Get the first application element of the application
   * relation.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The first application element of the application
   *          relation.
   */
   ApplicationElement getElem1()
      raises (AoException);

   /* (6003)
   * Get the second application element of the application
   * relation.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The second application element of the application
   *          relation.
   */
   ApplicationElement getElem2()
      raises (AoException);

   /* (6004)
   * Get the inverse relation range of the application
   * relation.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The inverse relation range of the application
   *          relation.
   */
   RelationRange getInverseRelationRange()
      raises (AoException);

   /* (6005)
   * Get the inverse relationship of the application relation.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The inverse relationship of the application
   *          relation.
   */
   Relationship getInverseRelationship()
      raises (AoException);

   /* (6006)
   * Get the name of the application relation.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The name of the application relation.
   */
   Name getRelationName()
      raises (AoException);

   /* (6007)
   * Get the relation range of the application relation.
   *
```

```
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The relation range of the application relation.
*/
RelationRange getRelationRange()
   raises (AoException);

/* (6008)
* Get the relationship of the application relation.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The relationship of the application relation.
*/
Relationship getRelationship()
   raises (AoException);

/* (6009)
* Get the relation type of the application relation.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The relation type of the application relation.
*/
RelationType getRelationType()
   raises (AoException);

/* (6010)
```

```
* Set the base relation of the application relation.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The relation type and relation range is copied from the
* base relation. The previous values get lost.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_RELATION
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  baseRel  The base relation.
*/
void setBaseRelation(
   in BaseRelation baseRel)
   raises (AoException);

/* (6011)
* Set the first application element of the application
* relation.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_ELEMENT
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  applElem  The application element.
*/
void setElem1(
   in ApplicationElement applElem)
   raises (AoException);
```

```
/* (6012)
* Set the second application element of the application
* relation.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_ELEMENT
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  applElem  The application element.
*/
void setElem2(
   in ApplicationElement applElem)
   raises (AoException);

/* (6013)
* Set the relation range of an application relation.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* It is only allowed to set the relation type if no base
* relation is defined.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_HAS_BASE_RELATION
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_RELATION_RANGE
*    AO_IS_BASE_RELATION
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  arRelationRange  The inverse relation range.
```

```
*/
void setInverseRelationRange(
   in RelationRange arRelationRange)
   raises (AoException);

/* (6014)
* Set the name of an application relation.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The name of an application attribute must not exceed the
* maximum name length of the underlying physical storage.
* The current physical storage specification restricts it to 30
* characters.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  arName  The application relation name.
*/
void setRelationName(
   in Name arName)
   raises (AoException);

/* (6015)
* Set the relation range of an application relation.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* It is only allowed to set the relation type if no base
* relation is defined.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_HAS_BASE_RELATION
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_RELATION_RANGE
*    AO_IS_BASE_RELATION
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  arRelationRange  The relation range.
*/
void setRelationRange(
   in RelationRange arRelationRange)
   raises (AoException);

/* (6016)
* Set the relation type of an application relation.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The relationship is automatically set when the relation
* type is set. It is only allowed to set the relation type
* if no base relation is defined.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_HAS_BASE_RELATION
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_RELATION_TYPE
*    AO_IS_BASE_RELATION
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  arRelationType  The relation type.
*/
void setRelationType(
   in RelationType arRelationType)
   raises (AoException);

/* (6017)
* Get the inverse name of the application relation. The
* inverse name of an application relation is the name of the
* relation seen from the other application element.
*
* @throws AoException
```

```
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The inverse name of the application relation.
   */
   Name getInverseRelationName()
      raises (AoException);

   /* (6018)
   * Set the name of an application relation.
   *
   * It is allowed to modify the object outside a transaction
   * but it is recommended to activate a transaction.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  arInvName  The inverse application relation name.
   */
   void setInverseRelationName(
      in Name arInvName)
      raises (AoException);

}; // Interface ApplicationRelation.

/*
* The ASAM ODS application model interface.
*/
interface ApplicationStructure {

   /* (7001)
   * Check the application model for ASAM ODS conformity. The
   * first error found is reported by an exception. The
   * following checks are performed:
   *
   *  - Each application element must be derived from a valid
   *    base element.
   *  - An application attribute is derived from one base
   *    attribute. It is not allowed to derive more than one
   *    application attribute from the same base attribute. It is
   *    allowed that application attributes are not derived from
   *    any base attribute.
   *  - All application elements must have at least the
   *    mandatory attributes.
   *  - Each application elements must be identified by a
   *    unique Asam path. No "floating" application elements are
   *    allowed.
   *  - All relations required by the base model must be
   *    present.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_DUPLICATE_BASE_ATTRIBUTE
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_INVALID_RELATION
   *    AO_MISSING_ATTRIBUTE
   *    AO_MISSING_RELATION
   *    AO_MISSING_APPLICATION_ELEMENT
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_PATH_TO_ELEMENT
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   */
   void check()
      raises (AoException);

   /* (7002)
   * Create a new application element in the application model.
   *
   *
   * It is allowed to modify the object outside a transaction
   * but it is recommended to activate a transaction.
   *
   * The information whether or not the new application element
   * is a top level element is taken from the specified base
   * element. The Id of the application element is set
   * automatically. The mandatory base attributes are created
   * automatically. Optional attributes have to be created by
   * the calling program. The application attribute interface
   * methods may be used to modify the attributes.
   *
```

```
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  baseElem  The base element from which the
*                   application element is derived.
*
* @return  The new application element.
*/
ApplicationElement createElement(
   in BaseElement baseElem)
   raises (AoException);

/* (7003)
* Create a new relation.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The relation is part of the application model. The
* application relation interface methods may be used to
* modify the relation.
*
* The default properties of a new application relation
* are:
*    BaseRelation   NULL
*    Element1       NULL
*    Element2       NULL
*    Range          -2, -2
*    Name           NULL
*    Type           INFO
* When element 1 or element 2 is set before the name of the
* relation is specified, the name of the application
* relation is set to "AUTOGEN".
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The new application relation.
*/
ApplicationRelation createRelation()
   raises (AoException);

/* (7004)
* Get the application element with the requested Id.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aeId  The Id of the requested application element.
*
* @return  The requested application element.
*/
ApplicationElement getElementById(
   in T_LONGLONG aeId)
   raises (AoException);

/* (7005)
* Get the application element with the requested name.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aeName  The name of the requested application
*                 element.
*
* @return  The requested application element.
*/
ApplicationElement getElementByName(
   in Name aeName)
   raises (AoException);

/* (7006)
* Get the application elements whose names match the
* pattern. The pattern is case sensitive and may contain
* wildcard characters.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aePattern  The name or the search pattern for the
*                    requested application elements.
*
* @return  The requested application elements.
*/
ApplicationElementSequence getElements(
   in Pattern aePattern)
   raises (AoException);

/* (7007)
* Get the names of application elements that are derived
* from the specified base element.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_BASETYPE
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aeType  The requested base element type. The base
*                 element type can be a pattern.
*
* @return  The requested application element names.
*/
ApplicationElementSequence getElementsByBaseType(
   in BaseType aeType)
   raises (AoException);

/* (7008)
* Get the instance element specified by the ASAM path.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_ASAM_PATH
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  asamPath  The ASAM path of the requested instance
*                   element.
*
* @return  The requested instance element.
*/
InstanceElement getInstanceByAsamPath(
   in Name asamPath)
   raises (AoException);

/* (7009)
* Returns the relations between two application elements.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  applElem1  The first application element.
*
* @param  applElem2  The second application element.
*
* @return  The relations between the specified application
*          elements.
*/
ApplicationRelationSequence getRelations(
   in ApplicationElement applElem1,
   in ApplicationElement applElem2)
   raises (AoException);

/* (7010)
* Get the top level application elements which are inherted
* from the base element that matches the base type. If the
* given base type is no top level base element an exception
* is thrown. A top level application element is an
* application element without a father.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_BASETYPE
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aeType  The requested base type. The base element
*                 type can be a pattern.
*
* @return  The top level application elements.
*/
ApplicationElementSequence getTopLevelElements(
   in BaseType aeType)
   raises (AoException);

/* (7011)
* Get the names of the application elements that  match the
* pattern. The pattern is case sensitive and may contain
* wildcard characters.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aePattern  The name or the search pattern for the
*                    requested base elements.
*
* @return  The names of the application elements.
```

```
*/
NameSequence listElements(
   in Pattern aePattern)
   raises (AoException);

/* (7012)
* Get the names of application elements that are  derived
* from the given base type.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_BASETYPE
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aeType  The requested base type. The base element
*                 type can be a pattern.
*
* @return  The names of the application elements.
*/
NameSequence listElementsByBaseType(
   in BaseType aeType)
   raises (AoException);

/* (7013)
* Get the names of the top level application elements that
* are derived from the given base type. If the given base
* type is not a top level base element an exception is
* thrown. A top level application element is an application
* element without a father.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_BASETYPE
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aeType  The requested base type.
```

```
*
* @return  The names of the application elements.
*/
NameSequence listTopLevelElements(
   in BaseType aeType)
   raises (AoException);

/* (7014)
* Remove an application element from the application model.
*
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
*   - Only allowed:
*       - if the application element is empty
*         (has no instances).
*       - no relations with other application elements.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_HAS_INSTANCES
*    AO_HAS_REFERENCES
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_FOUND
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  applElem  The application element to be removed.
*/
void removeElement(
   in ApplicationElement applElem)
   raises (AoException);

/* (7015)
* This method removes an application relation from the
* application model.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The elements of the relation are still part of the
* application model. If there are instances of the relation
* they are also removed.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_HAS_INSTANCES
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_FOUND
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  applRel  The application relation to be removed.
*/
void removeRelation(
   in ApplicationRelation applRel)
   raises (AoException);

/* (7016)
* Get the instance elements specified by the element id.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_ASAM_PATH
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  ieIds  The sequence with the element id.
*
* @return  The requested instance element sequence.
*/
InstanceElementSequence getInstancesById(
   in ElemIdSequence ieIds)
   raises (AoException);

/* (7017)
* Get the current client session in which the application
* model is created.
*
* @throws AoException
* with the following possible error codes:
```

```
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @return  The current client session.
*/
AoSession getSession()
   raises (AoException);

/* (7018)
* Create a new enumeration definition. This method modifies
* the application model and is only allowed for the
* superuser.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*    AO_ACCESS_DENIED
*
* @param  enumName  Name of the enumeration
*
* @return  The new created enumeration
*/
EnumerationDefinition createEnumerationDefinition(
   in T_STRING enumName)
   raises (AoException);

/* (7019)
* Remove the enumeration definition. The server checks if
* the enumeration is still in use by one of the attributes.
* This method modifies the application model and is only
* allowed for the superuser.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*    AO_ACCESS_DENIED
*
* @param  enumName  Name of the enumeration to remove.
*/
void removeEnumerationDefinition(
   in T_STRING enumName)
   raises (AoException);

/* (7020)
* Get the list of all enumeration names.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @return  List with all enumeration names.
*/
NameSequence listEnumerations()
   raises (AoException);

/* (7021)
* Get the specified enumeration definition.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  enumName  Name of the requested enumeration.
```

```
   *
   * @return  The enumeration definition.
   */
   EnumerationDefinition getEnumerationDefinition(
      in T_STRING enumName)
      raises (AoException);

   /* (7022)
   * Create the relation between a list of instances. The
   * number of instances in both list must be identical. The
   * application element of the instances in each list must be
   * identical. The application elements must match the
   * application elements of the application relation. The
   * index in the list of the instances defines related
   * instances.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *    AO_TRANSACTION_NOT_ACTIVE
   *    AO_INVALID_REQUEST
   *
   * @param  applRel  The application relation.
   *
   * @param  elemList1  The list with the instances of one
   *                    application element for which the
   *                    relation will be created.
   *
   * @param  elemList2  The list with the related instances.
   */
   void createInstanceRelations(
      in ApplicationRelation applRel,
      in InstanceElementSequence elemList1,
      in InstanceElementSequence elemList2)
      raises (AoException);

}; // Interface ApplicationStructure.

/*
* The ASAM ODS base attribute interface.
*/
```

```
interface BaseAttribute {

   /* (8001)
   * Get the data type of the base attribute.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The data type of the base attribute.
   */
   DataType getDataType()
      raises (AoException);

   /* (8002)
   * Get the name of the base attribute.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The name of the base attribute.
   */
   Name getName()
      raises (AoException);

   /* (8003)
   * Get the obligatory flag of the base attribute.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The obligatory flag of the base attribute.
   */
   T_BOOLEAN isObligatory()
      raises (AoException);

   /* (8004)
   * Get the unique flag of the base attribute.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The unique flag of the base attribute.
   */
   T_BOOLEAN isUnique()
      raises (AoException);

   /* (8005)
   * Return the base element to which the attibute belongs..
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The base element of the attribute.
   */
   BaseElement getBaseElement()
      raises (AoException);

}; // Interface BaseAttribute.

/*
* The ASAM ODS base element interface.
*/
interface BaseElement {

   /* (9001)
   * Get all known relations of the base element.
```

```
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  All known relations of the base element.
*/
BaseRelationSequence getAllRelations()
   raises (AoException);

/* (9002)
* Get attributes of the base element.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  baPattern  The name or the search pattern for the
*                    requested base attributes.
*
* @return  The requested attributes of the base element.
*/
BaseAttributeSequence getAttributes(
   in Pattern baPattern)
   raises (AoException);

/* (9003)
* Get the related elements of a base element defined by the
* relationship.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_RELATIONSHIP
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  brRelationship  The requested relationship.
*
* @return  The related elements of a base element.
*/
BaseElementSequence getRelatedElementsByRelationship(
   in Relationship brRelationship)
   raises (AoException);

/* (9004)
* Get the base element's relations of the requested relation
* type.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_RELATION_TYPE
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  brRelationType  The requested relation type.
*
* @return  The base element's relations of the requested
*          type.
*/
BaseRelationSequence getRelationsByType(
   in RelationType brRelationType)
   raises (AoException);

/* (9005)
* Get the type of the base element. The type of the base
* element is identical with the name of the base element.
* The type of the base element is a string.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
```

```
*
* @return  The type of the base element.
*/
BaseType getType()
   raises (AoException);

/* (9006)
* Get whether or not the base element is a top level
* element. Top level elements are elements without a father.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  Boolean whether or not the base element is a top
*          level element.
*/
T_BOOLEAN isTopLevel()
   raises (AoException);

/* (9007)
* Get attribute names of the base element.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  baPattern  The name or the search pattern for the
*                    requested base attribute names.
*
* @return  The requested attribute names of the base
*          element.
*/
NameSequence listAttributes(
   in Pattern baPattern)
   raises (AoException);
   /* (9008)
   * Get the related element names of the base element defined
   * by the relationship.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_INVALID_RELATIONSHIP
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  brRelationship  The requested relationship.
   *
   * @return  The related element names of the base element.
   */
   BaseTypeSequence listRelatedElementsByRelationship(
      in Relationship brRelationship)
      raises (AoException);

}; // Interface BaseElement.

/*
* The ASAM ODS base relation interface.
*/
interface BaseRelation {

   /* (10001)
   * Get the first base element of the base relation.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The first base element of the base relation.
   */
   BaseElement getElem1()
      raises (AoException);

   /* (10002)
```

```
* Get the second base element of the base relation.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The second base element of the base relation.
*/
BaseElement getElem2()
   raises (AoException);

/* (10003)
* Get the inverse relation range of the base relation.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The inverse relation range of the base relation.
*/
RelationRange getInverseRelationRange()
   raises (AoException);

/* (10004)
* Get the inverse relationship of the base relation.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The inverse relationship of the base relation.
*/
Relationship getInverseRelationship()
   raises (AoException);
```

```
/* (10005)
* Get the relation name of the base relation.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The relation name of the base relation.
*/
Name getRelationName()
   raises (AoException);

/* (10006)
* Get the relation range of the base relation.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The relation range of the base relation.
*/
RelationRange getRelationRange()
   raises (AoException);

/* (10007)
* Get the relationship of the base relation.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The relationhip of the base relation.
*/
```

```
   Relationship getRelationship()
      raises (AoException);

   /* (10008)
   * Get the relation type of the base relation.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The relation type of the base relation.
   */
   RelationType getRelationType()
      raises (AoException);

}; // Interface BaseRelation.

/*
* The ASAM ODS base model interface.
*/
interface BaseStructure {

   /* (11001)
   * Get the base element that matches the requested type. The
   * type of a base element is identical with the name of the
   * base element.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_INVALID_BASETYPE
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  beType  The name of the requested base element.
   *
   * @return  The requested base element.
   */
   BaseElement getElementByType(
      in BaseType beType)
      raises (AoException);

   /* (11002)
   * Get the base elements that match the pattern. The pattern
   * is case sensitive and may contain wildcard characters.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  bePattern  The name or the search pattern for the
   *                    requested base elements.
   *
   * @return  The requested base elements.
   */
   BaseElementSequence getElements(
      in Pattern bePattern)
      raises (AoException);

   /* (11003)
   * Get the base relation between two base elements.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  elem1  The base element from which the relation
   *                starts.
   *
   * @param  elem2  The base element to which the relation
   *                points.
   *
   * @return  The base relation between the two base elements.
   */
   BaseRelation getRelation(
      in BaseElement elem1,
      in BaseElement elem2)
      raises (AoException);

   /* (11004)
   * Get the top level base elements that match the pattern.
   * The pattern is case sensitive and may contain wildcard
   * characters. A top level base element is a base element
   * without a father.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  bePattern  The name or the search pattern for the
   *                    requested top level base elements.
   *
   * @return  The requested top level base elements.
   */
   BaseElementSequence getTopLevelElements(
      in Pattern bePattern)
      raises (AoException);

   /* (11005)
   * Get the version of the base model.The version of the base
   * model is the version of the ASAM ODS base model.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The version of the ASAM ODS base model.
   */
   T_STRING getVersion()
      raises (AoException);

   /* (11006)
```

```
* Get the base element names that match the pattern. The
* pattern is case sensitive and may contain wildcard
* characters.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  bePattern  The name or the search pattern for the
*                    requested base element names.
*
* @return  The requested base element names.
*/
BaseTypeSequence listElements(
   in Pattern bePattern)
   raises (AoException);

/* (11007)
* Get the top level base element names that match the
* pattern. The pattern is case sensitive and may contain
* wildcard characters. A top level base element is a base
* element without a father.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  bePattern  The name or the search pattern for the
*                    requested top level base element names.
*
* @return  The requested top level base element names.
*/
BaseTypeSequence listTopLevelElements(
   in Pattern bePattern)
   raises (AoException);
```

```
}; // Interface BaseStructure.

/*
* The ASAM ODS blob interface.
*/
interface Blob {

   /* (12001)
   * Append a byte sequence to the binary large object.
   *
   * It is allowed to modify the object outside a transaction
   * but it is recommended to activate a transaction.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  value  The byte sequence.
   */
   void append(
      in S_BYTE value)
      raises (AoException);

   /* (12002)
   * Compares the content of the binary large object. The
   * headers are not compared.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  aBlob  The blob to compare.
   *
   * @return  A flag whether or not the compared blobs are
   *          equal.
   */
```

```
T_BOOLEAN compare(
   in T_BLOB aBlob)
   raises (AoException);

/* (12003)
* Get a part of the binary large object.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  offset  The starting position of the data in the
*                 blob.
*
* @param  length  The number of bytes requested from the
*                 blob.
*
* @return  The request part of the blob data.
*/
S_BYTE get(
   in T_LONG offset,
   in T_LONG length)
   raises (AoException);

/* (12004)
* Get the header of the binary large object.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The blob header.
*/
T_STRING getHeader()
   raises (AoException);

/* (12005)
```

```
* Get the length of the binary large object without loading
* it.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The blob length.
*/
T_LONG getLength()
   raises (AoException);

/* (12006)
* Clear the binary large object and set the new data.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  value  The new blob data.
*/
void set(
   in S_BYTE value)
   raises (AoException);

/* (12007)
* Set the header of a binary large object.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
```

```
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  header  The blob header.
   */
   void setHeader(
      in T_STRING header)
      raises (AoException);

   /* (12008)
   * Destroy the object on the server. The destructor of the
   * client, so the server knows this object is not used
   * anymore by the client. Access to this object after the
   * destroy method will lead to an exception.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *    AO_TRANSACTION_NOT_ACTIVE
   */
   void destroy()
      raises (AoException);

}; // Interface Blob.

/*
* The ASAM ODS column interface. It is not inherited from
* InstanceElement. Via the method getColumn of the interface
* SubMatrix the column is accessed. The column is only used for
* read access. With the creation of instances at the
* application element of type AoLocalColumn and the relation to
* the instances of the application element of type AoSubMatrix
* a local column can be created. The column name is used for
* the SMatLink and the column is used to store a formula for
* the calculation of the values of the column. There is no
* definition of the formula language at the moment.
*/
interface Column {
```

```
/* (13001)
* Get the formula of the column.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The formula of the column.
*/
T_STRING getFormula()
   raises (AoException);

/* (13002)
* Get the name of the column.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The name of the column.
*/
Name getName()
   raises (AoException);

/* (13003)
* Get the source measurement quantity.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The source measurement quantity.
*/
```

```
InstanceElement getSourceMQ()
   raises (AoException);

/* (13004)
* Get the unit of the column.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The unit of the column.
*/
T_STRING getUnit()
   raises (AoException);

/* (13005)
* Set the formula of the column.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  formula  The formula.
*/
void setFormula(
   in T_STRING formula)
   raises (AoException);

/* (13006)
* Set the unit of the column. This is only a temporary
* conversion unit when getting the data of a column. This
* unit is not stored in the database. If a permanent storage
* of the conversion unit is required the corresponding
* measurement quantity needs to be changed.
```

```
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  unit  The physical unit.
*/
void setUnit(
   in T_STRING unit)
   raises (AoException);

/* (13007)
* Is the column an independent column
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The independent flag of the column.
*/
T_BOOLEAN isIndependent()
   raises (AoException);

/* (13008)
* Is the column an scaling column
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  Tells if the column is a scaling column.
```

```
*/
T_BOOLEAN isScaling()
   raises (AoException);

/* (13009)
* Set the column as an indepent column.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  independent  The new value of the independent
*                      flag.
*/
void setIndependent(
   in T_BOOLEAN independent)
   raises (AoException);

/* (13010)
* Set the column to a scaling column.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  scaling  The new value of the scaling flag.
*/
void setScaling(
   in T_BOOLEAN scaling)
   raises (AoException);

/* (13011)
* Get the data type of the column.
*
```

```
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *    AO_TRANSACTION_NOT_ACTIVE
   *
   * @return  The data type of the column.
   */
   DataType getDataType()
      raises (AoException);

   /* (13012)
   * Destroy the object on the server. The destructor of the
   * client, so the server knows this object is not used anymore
   * by the client. Access to this object after the destroy
   * method will lead to an exception.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *    AO_TRANSACTION_NOT_ACTIVE
   */
   void destroy()
      raises (AoException);

}; // Interface Column.

/*
* The ASAM ODS instance element interface. It is allowed to
* modify the instances outside a transaction but recommended to
* activate a transaction before the instances are modified. The
* modifications to instances get permanent when the transaction
* is committed.
*/
interface InstanceElement {

   /* (14001)
```

```
* Add an instance attribute to the instance. The instance
* attribute is built as a Name/Value/Unit tuple on the
* client. This method has to copy the data from the client
* to the server. The name of the attribute must be unique.
*
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  instAttr  The instance attribute to be added.
*/
void addInstanceAttribute(
   in NameValueUnit instAttr)
   raises (AoException);

/* (14002)
* Create a relation between the current and the given
* instance. Check if the application elements of the
* relation matches the application elements of the
* instances.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_RELATION
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  relation  The application relation.
*
* @param  instElem  The instance element.
```

```
*/
void createRelation(
   in ApplicationRelation relation,
   in InstanceElement instElem)
   raises (AoException);

/* (14003)
* Get the application element of the instance element.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The application element from which the instance
*          element is derived.
*/
ApplicationElement getApplicationElement()
   raises (AoException);

/* (14004)
* Get the ASAM-Path of the instance element.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The ASAM path to the instance element.
*/
Name getAsamPath()
   raises (AoException);

/* (14005)
* Get the Id of the instance element.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The Id of the instance element.
*/
T_LONGLONG getId()
   raises (AoException);

/* (14006)
* Get the name of the instance element.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The name of the instance element.
*/
Name getName()
   raises (AoException);

/* (14007)
* Get the related instances. The application relation and
* the name of the related  instances specify the listed
* instances. The pattern is case sensitive and may contain
* wildcard characters.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_RELATION
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  applRel  The application relation.
*
* @param  iePattern  The name or the search pattern for the
*                    related instance names.
*
```

```
* @return  The related instances.
*/
InstanceElementIterator getRelatedInstances(
   in ApplicationRelation applRel,
   in Pattern iePattern)
   raises (AoException);

/* (14008)
* Get the list of related instances. The relationship and
* the name of the related instances specify the listed
* instances. The pattern is case sensitive and may contain
* wildcard characters.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_RELATIONSHIP
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  ieRelationship  The requested relationship.
*
* @param  iePattern  The name or the search pattern for the
*                    related instance names.
*
* @return  The related instances.
*/
InstanceElementIterator getRelatedInstancesByRelationship(
   in Relationship ieRelationship,
   in Pattern iePattern)
   raises (AoException);

/* (14009)
* Get the attribute value (name, value and unit) of the
* given attribute of the instance element. This method will
* not return the value of relation attributes, use the
* method getRelatedInstances.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  attrName  The name of the requested attribute.
*
* @return  The attribute value.
*/
NameValueUnit getValue(
   in Name attrName)
   raises (AoException);

/* (14010)
* Get the attribute value (value and unit) of the attribute
* inherited from the given base attribute of the instance
* element. The base name is case insensitive and may not
* contain wildcard characters.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  baseAttrName  The base name of the requested
*                       attribute.
*
* @return  The attribute value.
*/
NameValueUnit getValueByBaseName(
   in Name baseAttrName)
   raises (AoException);

/* (14011)
* Get the attribute names from the instance element. The
* attributes reserved for a relation are not listed.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_ATTRIBUTE_TYPE
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  iaPattern  The name or the search pattern for the
*                    attribute names.
*
* @param  aType  The requested attribute type.
*
* @return  The names of the attributes.
*/
NameSequence listAttributes(
   in Pattern iaPattern,
   in AttrType aType)
   raises (AoException);

/* (14012)
* Get the names of the related instances. The application
* relation and the name of the related instances specifies
* the listed names. The pattern is case sensitive and may
* contain wildcard characters.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  ieRelation  The application relation.
*
* @param  iePattern  The name or the search pattern for the
*                    related instance names.
*
* @return  The names of the related instances.
*/
NameIterator listRelatedInstances(
   in ApplicationRelation ieRelation,
   in Pattern iePattern)
   raises (AoException);

/* (14013)
* Get the names of the related instances. The relationship
* and the name of the related instances specify the listed
* names. The pattern is case sensitive and may contain
* wildcard characters.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_RELATIONSHIP
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  ieRelationship  The requested relationship.
*
* @param  iePattern  The name or the search pattern for the
*                    related instance names.
*
* @return  The names of the related instances.
*/
NameIterator listRelatedInstancesByRelationship(
   in Relationship ieRelationship,
   in Pattern iePattern)
   raises (AoException);

/* (14014)
* Remove an instance attribute.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The application attributes can't be removed.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_FOUND
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  attrName  The name of the attribute to be removed.
*/
void removeInstanceAttribute(
   in Name attrName)
   raises (AoException);

/* (14015)
* Remove the relation between the current instance and the
* given instance. It is necessary to specify the instance
* element in case of n:m relations if not all relations
* shall be deleted.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_RELATION
*    AO_NOT_FOUND
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  applRel  The relation to be removed.
*
* @param  instElem_nm  The instance element for specific
*                      delete from n:m relations.
*/
void removeRelation(
   in ApplicationRelation applRel,
   in InstanceElement instElem_nm)
   raises (AoException);

/* (14016)
* Rename the instance attribute. The application attributes
* can't be renamed.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_DUPLICATE_NAME
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_FOUND
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  oldName  The old instance attribute name.
*
* @param  newName  The new instance attribute name.
*/
void renameInstanceAttribute(
   in Name oldName,
   in Name newName)
   raises (AoException);

/* (14017)
* Set the name of an instance element.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  iaName  The instance attribute name.
*/
void setName(
   in Name iaName)
   raises (AoException);

/* (14018)
* Set the value of an application or instance attribute.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The name of the attribute is specified by the name of the
* NameValueUnit tuple. If the application attribute flag
* unique is set, the uniqueness of the new value is
* checked.
*
```

```
* This method can not be used to set the value of a relation
* attribute, use the method createRelation.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_DUPLICATE_VALUE
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  value  The value to be set in the instance
*                element.
*/
void setValue(
   in NameValueUnit value)
   raises (AoException);

/* (14019)
* Cast an instance element to a measurement. There are some
* object-oriented languages which do not allow this cast.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_INVALID_BASETYPE
*
* @return  The instance of type measurement.
*/
Measurement upcastMeasurement()
   raises (AoException);

/* (14020)
* Cast an instance element to a submatrix. There are some
* object-oriented languages which do not allow this cast.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_INVALID_BASETYPE
*
* @return  The instance of type submatrix.
*/
SubMatrix upcastSubMatrix()
   raises (AoException);

/* (14021)
* Get the attribute value (name, value and unit) of the
* given attribute of the instance element.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_INCOMPATIBLE_UNITS
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  attr  The name of the requested attribute and the
*               unit of the attribute value.
*
* @return  The attribute value, value converted to the
*          requested unit.
*/
NameValueUnit getValueInUnit(
   in NameUnit attr)
   raises (AoException);

/* (14022)
* Set a sequences of values of an application or instance
* attributes.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The name of the attribute is specified by the name of the
* NameValueUnit tuple. If the application attribute flag
* unique is set, the uniqueness of the new value is checked.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_DUPLICATE_VALUE
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  values  The sequence of the values to be set at
*                 the instance element.
*/
void setValueSeq(
   in NameValueUnitSequence values)
   raises (AoException);

/* (14023)
* The given usergroup the rights should be set for. <rights>
* defines the rights to set or to clear. If the parameter
* <set> is set to 'set', the rights in <rights> are set, all
* others are cleared. If the parameter <set> is set to
* 'add', the rights in <rights> are added to the existing
* rights. If the parameter <set> is set to 'remove', the
* rights in <rights> are removed from the existing rights.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  usergroup  The usergroup for which the rights will
*                    be modified.
*
* @param  rights  The new right for the usergroup. The
*                 rights constants are defined in the
*                 interface SecurityRights. The interface
*                 definition language IDL does not allow to
*                 set the values of enumerations thus the
*                 constant definitions had to be done in an
*                 interface.
*
```

```
* @param  set  What to do with the new right.
*/
void setRights(
   in InstanceElement usergroup,
   in T_LONG rights,
   in RightsSet set)
   raises (AoException);

/* (14024)
* Retrieve access control list information of the given
* object.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @return  The access control list entries of the given
*          application element.
*/
ACLSequence getRights()
   raises (AoException);

/* (14025)
* Retrieve access control list information for the initial
* rights of the given object.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @return  The access control list entries with the initial
*          rights of the given application element.
*/
InitialRightSequence getInitialRights()
```

```
   raises (AoException);

/* (14026)
* The given usergroup the initial rights should be set for.
* <rights> defines the rights to set or to clear. If the
* parameter <set> is set to 'set', the rights in <rights>
* are set, all others are cleared. If the parameter <set> is
* set to 'add', the rights in <rights> are added to the
* existing rights. If the parameter <set> is set to
* 'remove', the rights in <rights> are removed from the
* existing rights.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  usergroup  The usergroup for which the initial
*                    rights will be modified.
*
* @param  rights  The new initial rights for the usergroup.
*                 The rights constants are defined in the
*                 interface SecurityRights. The interface
*                 definition language IDL does not allow to
*                 set the values of enumerations thus the
*                 constant definitions have to be done in
*                 the interface.
*
* @param  refAid  The Id of referencing application element
*                 for which the initial rights will be used.
*                 If no refAid is set the initial rights
*                 will be used for each new instance element
*                 independent of the application element.
*
* @param  set  What to do with the new initial rights.
*/
void setInitialRights(
   in InstanceElement usergroup,
   in T_LONG rights,
   in T_LONGLONG refAid,
   in RightsSet set)
   raises (AoException);

/* (14027)
* Provides an easy-to-use and effective copy mechanismn for
* instance elements inside the server. The new instance
* elements gets a copy of all attribute values and
* informational relations that are available in the original
* instance element. The new instance element has the same
* parent as the original instance element but it does not
* have references to any children of the original instance
* element.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  newName  The name of the new instance element. If
*                  a new version shall be created this
*                  parameter may be NULL to use the same
*                  name for the copy. In this case a new
*                  version must be provided.
*
* @param  newVersion  The version of the new instance
*                     element. This parameter may be NULL if
*                     a new name is provided.
*
* @return  The reference to the copied instance element.
*/
InstanceElement shallowCopy(
   in T_STRING newName,
   in T_STRING newVersion)
   raises (AoException);

/* (14028)
* Provides an easy-to-use and effective copy mechanismn for
* instance element hierarchies inside the server (e.g. copy
* a project with all tests or copy a test with all
* measurements). The deep copy follows only the child
* references but not the informational references. Example:
* Copying elements of type AoMeasurement does not include
* copying the referenced elements of type
* AoMeasurementQuantity. The copied instance elements of
* type AoMeasurement will reference the same measurement
* quantities as the original. An application that wants to
* copy the measurement quantity also must do this (including
* setting the proper references) by itself e.g. with another
* call to shallowCopy; deepCopy is not necessary in this
* case because AoMeasurementQuantity has no children.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  newName  The name of the new instance element. If
*                  a new version shall be created this
*                  parameter may be NULL to use the same
*                  name for the copy. In this case a new
*                  version must be provided.
*
* @param  newVersion  The version of the new instance
*                     element. This parameter may be NULL if
*                     a new name is provided.
*
* @return  The reference to the copied instance element
*          hierarchy.
*/
InstanceElement deepCopy(
   in T_STRING newName,
   in T_STRING newVersion)
   raises (AoException);

/* (14029)
* Get the sequence of the values of the application or
* instance attributes, specified by their names.  The name
* sequence can use a pattern (*) for all attributes of the
* instance element. This means that application as well as
* instance attributes will be delivered.
*
* @throws AoException
* with the following possible error codes:
```

```
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  attrNames  The names of the attributes to be
*                    reported.
*
* @return  The sequence of the attribute values.
*/
NameValueUnitSequence getValueSeq(
   in NameSequence attrNames)
   raises (AoException);

/* (14030)
* Destroy the object on the server. The destructor of the
* client, so the server knows this object is not used anymore
* by the client. Access to this object after the destroy
* method will lead to an exception.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*/
void destroy()
   raises (AoException);

/* (14031)
* Compare two instance elements. The Ids of the application
* elements and the Ids of the instance elements are
* compared. The Ids of the application element will be
* compared first.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  compIeObj  The instance element to compare with.
*
* @return  The difference of the Id's. Meaning:
*          diff < 0 ElemId of instance is smaller then
*          instance to compare with.
*          diff == 0 ElemId is identical.
*          diff > 0 ElemId of instance is greater then
*          instance to compare with.
*/
T_LONGLONG compare(
   in InstanceElement compIeObj)
   raises (AoException);

/* (14032)
* Create a list with instances which are related to the
* actual instance element. The attribute are given with the
* name of the sequence. The values of the attributes are
* given in the value sequence. The index in the different
* value sequences match for one instance element. The index
* in the instance element sequence of the related instances
* match for the instance with the same index in the value
* sequence.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*    AO_INVALID_REQUEST
*
* @param  applRel  The application relation for wich the
*                  related instances will be created.
*
* @param  attributes  The attributes of the new created
*                     instances.
*
* @param  relatedInstances  The list with related instances
*                           for different application
```

```
   *                           relations.
   *
   * @return  The list with the new created instances.
   */
   InstanceElementSequence createRelatedInstances(
      in ApplicationRelation applRel,
      in NameValueSeqUnitSequence attributes,
      in ApplicationRelationInstanceElementSeqSequence relatedInstances)
      raises (AoException);

}; // Interface InstanceElement.

/*
* The ASAM ODS instance element iterator interface.
*/
interface InstanceElementIterator {

   /* (15001)
   * Destroy the iterator and free the associated memory.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   */
   void destroy()
      raises (AoException);

   /* (15002)
   * Get the total number of elements accessible by the
   * iterator.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The number of elements accessible by the
   *          iterator.
   */
```

```
T_LONG getCount()
   raises (AoException);

/* (15003)
* Get the next n elements from the sequence.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  how_many  The number of requested elements.
*
* @return  The next n instance elements from the instance
*          sequence.
*/
InstanceElementSequence nextN(
   in T_LONG how_many)
   raises (AoException);

/* (15004)
* Get the next element from the sequence.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The next instance element from the instance
*          sequence.
*/
InstanceElement nextOne()
   raises (AoException);

/* (15005)
* Reset the pointer in the element sequence to the first
* element.
*
* @throws AoException
```

```
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   */
   void reset()
      raises (AoException);

}; // Interface InstanceElementIterator.

/*
* The ASAM ODS measurement interface.
*/
interface Measurement : InstanceElement {

   /* (16001)
   * Create a submatrix link. The submatrix link is only valid
   * in the current session. When the session is closed the
   * submatrix link will be destroyed.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The new submatrix link.
   */
   SMatLink createSMatLink()
      raises (AoException);

   /* (16002)
   * Get the list of the submatrix links .
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
```

```
      * @return  The available submatrix links.
      */
      SMatLinkSequence getSMatLinks()
         raises (AoException);

      /* (16003)
      * Get the value matrix of a measurement.
      *
      * @throws AoException
      * with the following possible error codes:
      *    AO_CONNECTION_LOST
      *    AO_IMPLEMENTATION_PROBLEM
      *    AO_NOT_IMPLEMENTED
      *    AO_NO_MEMORY
      *    AO_SESSION_NOT_ACTIVE
      *
      * @return  The value matrix.
      */
      ValueMatrix getValueMatrix()
         raises (AoException);

      /* (16004)
      * Remove a submatrix link.
      *
      * @throws AoException
      * with the following possible error codes:
      *    AO_BAD_PARAMETER
      *    AO_CONNECTION_LOST
      *    AO_IMPLEMENTATION_PROBLEM
      *    AO_NOT_FOUND
      *    AO_NOT_IMPLEMENTED
      *    AO_NO_MEMORY
      *    AO_SESSION_NOT_ACTIVE
      *
      * @param  smLink  The submatrix link to be removed.
      */
      void removeSMatLink(
         in SMatLink smLink)
         raises (AoException);

   }; // Interface Measurement.

   /*
   * The ASAM ODS name iterator interface.
   */
   interface NameIterator {
```

```
/* (17001)
* Destroy the iterator and free the associated memory.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*/
void destroy()
   raises (AoException);

/* (17002)
* Get the total number of elements accessible by the
* iterator.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The number of elements accessible by the
*          iterator.
*/
T_LONG getCount()
   raises (AoException);

/* (17003)
* Get the next n elements from the sequence.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  how_many  The number of requested elements.
```

```
   *
   * @return  The next n names from the name sequence.
   */
   NameSequence nextN(
      in T_LONG how_many)
      raises (AoException);

   /* (17004)
   * Get the next element from the sequence.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The next name from the name sequence.
   */
   Name nextOne()
      raises (AoException);

   /* (17005)
   * Reset the pointer in the element sequence to the first
   * element.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   */
   void reset()
      raises (AoException);

}; // Interface NameIterator.

/*
* The ASAM ODS name-value iterator interface.
*/
interface NameValueIterator {

   /* (18001)
```

```
* Destroy the iterator and free the associated memory.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*/
void destroy()
   raises (AoException);

/* (18002)
* Get the total number of elements accessible by the
* iterator.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The number of elements accessible by the
*          iterator.
*/
T_LONG getCount()
   raises (AoException);

/* (18003)
* Get the next n elements from the sequence.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  how_many  The number of requested elements.
*
* @return  The next n name-value pairs from the name-value
```

```
   *          pair sequence.
   */
   NameValueSequence nextN(
      in T_LONG how_many)
      raises (AoException);

   /* (18004)
   * Get the next element from the sequence.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The next name-value pair from the name-value pair
   *          sequence.
   */
   NameValue nextOne()
      raises (AoException);

   /* (18005)
   * Reset the pointer in the element sequence to the first
   * element.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   */
   void reset()
      raises (AoException);

}; // Interface NameValueIterator.

/*
* The ASAM ODS name-value-unit iterator interface. This
* interface is identical with the NameValueUnitIdIterator,
* except the unit is given as a string insead of an Id.
*/
interface NameValueUnitIterator {
```

```
/* (19001)
* Destroy the iterator and free the associated memory.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*/
void destroy()
   raises (AoException);

/* (19002)
* Get the total number of elements accessible by the
* iterator.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The number of elements accessible by the
*          iterator.
*/
T_LONG getCount()
   raises (AoException);

/* (19003)
* Get the next n elements from the sequence.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  how_many  The number of requested elements.
```

```
   *
   * @return  The next n name-value-unit tuples from the
   *          name-value-unit tuple sequence.
   */
   NameValueUnitSequence nextN(
      in T_LONG how_many)
      raises (AoException);

   /* (19004)
   * Get the next element from the sequence.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The next name-value-unit tuple from the
   *          name-value-unit tuple sequence.
   */
   NameValueUnit nextOne()
      raises (AoException);

   /* (19005)
   * Reset the pointer in the element sequence to the first
   * element.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   */
   void reset()
      raises (AoException);

}; // Interface NameValueUnitIterator.

/*
* The ASAM ODS submatrix link interface.
*/
interface SMatLink {
```

```
/* (20001)
* Get the link or build type.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The link type.
*/
BuildUpFunction getLinkType()
   raises (AoException);

/* (20002)
* Get the ordinal or sequence number
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The sequence number.
*/
T_LONG getOrdinalNumber()
   raises (AoException);

/* (20003)
* Get the first submatrix of the link.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The first submatrix of the link.
*/
```

```
SubMatrix getSMat1()
   raises (AoException);

/* (20004)
* Get the bind columns of the first submatrix used in the
* link (e.g. Time).
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The columns of the first submatrix.
*/
ColumnSequence getSMat1Columns()
   raises (AoException);

/* (20005)
* Get the second submatrix of the link.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The second submatrix of the link.
*/
SubMatrix getSMat2()
   raises (AoException);

/* (20006)
* Get the bind columns of the second submatrix used in the
* link (e.g. Time).
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The columns of the second submatrix.
*/
ColumnSequence getSMat2Columns()
   raises (AoException);

/* (20007)
* Set the build or link type.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_BUILDUP_FUNCTION
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  linkType  The requesed build-up function.
*/
void setLinkType(
   in BuildUpFunction linkType)
   raises (AoException);

/* (20008)
* Set the ordinal or sequence number.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_ORDINALNUMBER
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  ordinalNumber  The sequence number.
*/
void setOrdinalNumber(
   in T_LONG ordinalNumber)
   raises (AoException);

/* (20009)
```

```
* Set the first submatrix of the link.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_SUBMATRIX
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  subMat1  The first submatrix of the submatrix
*                  link.
*/
void setSMat1(
   in SubMatrix subMat1)
   raises (AoException);

/* (20010)
* Set the bind columns of the first submatrix used in the
* link (e.g. Time).
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_COLUMN
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  columns  The column sequence of the submatrix.
*/
void setSMat1Columns(
   in ColumnSequence columns)
   raises (AoException);

/* (20011)
* Set the second submatrix of the link.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
```

```
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_INVALID_SUBMATRIX
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  subMat2  The second submatrix of the submatrix
   *                  link.
   */
   void setSMat2(
      in SubMatrix subMat2)
      raises (AoException);

   /* (20012)
   * Set the bind columns of the second submatrix used in the
   * link (e.g. Time). If there is more than one column bound
   * the column sequence must be identical with the column
   * sequence of the first submatrix.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_INVALID_COLUMN
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  columns  The column sequence of the submatrix.
   */
   void setSMat2Columns(
      in ColumnSequence columns)
      raises (AoException);

}; // Interface SMatLink.

/*
* The ASAM ODS submatrix interface. The instances of the
* submatrix can be accessed via the method getRelatedInstances
* of the interface Measurement.
*/
interface SubMatrix : InstanceElement {

   /* (21001)
   * Get the columns of the submatrix. The column is not
   * inherited from the InstanceElement interface. This is the
   * only way to get a column. The columns are used in the
   * SMatLink interface to build the value matrix. The pattern
   * is case sensitive and may contain wildcard characters.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  colPattern  The name or the search pattern for the
   *                     column names.
   *
   * @return  The columns of the submatrix.
   */
   ColumnSequence getColumns(
      in Pattern colPattern)
      raises (AoException);

   /* (21002)
   * Get a value matrix of the submatrix.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_INVALID_SMATLINK
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The value matrix.
   */
   ValueMatrix getValueMatrix()
      raises (AoException);

   /* (21003)
   * Get the names of the columns of the submatrix. The name
   * sequence is identical with the names of the related
   * instances. The pattern is case sensitive and may contain
   * wildcard characters.
   *
```

```
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  colPattern  The name or the search pattern for the
   *                     column names.
   *
   * @return  The column names of the submatrix.
   */
   NameSequence listColumns(
      in Pattern colPattern)
      raises (AoException);

}; // Interface SubMatrix.

/*
* The ASAM ODS value matrix interface. Value matrix is an
* interface used by Measurement and SubMatrix to handle vectors
* of values.
*/
interface ValueMatrix {

   /* (22001)
   * Get the columns of the value matrix no matter whether the
   * column is dependent or independent. The pattern is case
   * sensitive and may contain wildcard characters.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  colPattern  The name or the search pattern for the
   *                     column names.
   *
   * @return  The columns of the value matrix, no matter
   *          whether the column is dependent, independent or
   *          scaling
   */
   ColumnSequence getColumns(
      in Pattern colPattern)
      raises (AoException);

   /* (22002)
   * Get the column count of the value matrix.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The number of columns of the value matrix.
   */
   T_LONG getColumnCount()
      raises (AoException);

   /* (22003)
   * Get the independent columns of the value matrix. The
   * independent columns are the columns used to build the
   * value matrix.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  colPattern  The name or the search pattern for the
   *                     independent column name.
   *
   * @return  The independent column of the value matrix.
   */
   ColumnSequence getIndependentColumns(
      in Pattern colPattern)
      raises (AoException);

   /* (22004)
```

```
* Get the row count of the value matrix.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The number of rows of the value matrix.
*/
T_LONG getRowCount()
   raises (AoException);

/* (22005)
* Get a measurement point of the value matrix. The parameter
* meaPoint specifies the row of the matrix. The iterator
* allows to access all elements in the row.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  meaPoint  The measurement point.
*
* @return  The requested measurement point.
*/
NameValueUnitIterator getValueMeaPoint(
   in T_LONG meaPoint)
   raises (AoException);

/* (22006)
* Get the values or a part of values of the column from the
* value matrix. The parameter column specifies from which
* column the values will be returned. The startPoint and
* pointCount specify the part of the vector. A startPoint =
* 0 and pointCount = rowCount will return the entire vector.
* When startPoint >= rowCount an exception is thrown. If
* startPoint + pointCount > rowCount only the remaining
* values of the vector are returned and no exception is
* thrown. Use the getName and getUnit method of the
* interface column for the name and the unit of the column.
* The name and the value are not stored at each element of
* the vector. The return type TS_ValueSeq is not a sequence
* of TS_Value but a special structure.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_COLUMN
*    AO_INVLAID_COUNT
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  col  The column to retrieve the values from.
*
* @param  startPoint  The starting point in the column.
*
* @param  count  The number of points to be retrieved.
*
* @return  The requested column values of the value matrix.
*/
TS_ValueSeq getValueVector(
   in Column col,
   in T_LONG startPoint,
   in T_LONG count)
   raises (AoException);

/* (22007)
* Get the names of the columns of the value matrix no matter
* whether the column is dependent or independent. The
* pattern is case sensitive and may contain wildcard
* characters.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
```

```
* @param  colPattern  The name or the search pattern for the
*                     column names.
*
* @return  The column names of the value matrix, no matter
*          whether the column is dependent, independent or
*          scaled by another one.
*/
NameSequence listColumns(
   in Pattern colPattern)
   raises (AoException);

/* (22008)
* Get the names of the independent columns of the
* value matrix. The independent columns are the columns used
* to build the value matrix.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  colPattern  The name or the search pattern for the
*                     independent column name.
*
* @return  The names of the independent columns of the value
*          matrix.
*/
NameSequence listIndependentColumns(
   in Pattern colPattern)
   raises (AoException);

/* (22009)
* Remove the values of the columns at a given measurement
* point. Remove the number of points of the given column. If
* the count is 0 all points until the end of the column are
* removed.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_COUNT
*    AO_NOT_FOUND
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  columnNames  The columns from which the
*                      measurement points are to be removed.
*
* @param  meaPoint  The measurement point to be removed.
*
* @param  count  The number of points to be removed from
*                each column.
*/
void removeValueMeaPoint(
   in NameSequence columnNames,
   in T_LONG meaPoint,
   in T_LONG count)
   raises (AoException);

/* (22010)
* Remove the values from a value vector. Beginning at
* startPoint the number of values specified in count are
* removed. If count is 0 all values from  the startPoint
* until the end of the vector are removed.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_COLUMN
*    AO_INVALID_COUNT
*    AO_NOT_FOUND
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  col  The column from which the values are to be
*              removed.
*
* @param  startPoint  The starting point for the value
*                     removal.
*
* @param  count  The number of points to be removed from the
*                column.
```

```
*/
void removeValueVector(
   in Column col,
   in T_LONG startPoint,
   in T_LONG count)
   raises (AoException);

/* (22011)
* Create or modify a measurement point of a value matrix.
* The sequence of name values specifies the names of the
* column with the new values.
* The meaning of the parameter setType is:
*
*    INSERT   Insert the values at meaPoint,
*             the current values at meaPoint
*             are moved to the end of
*             the new inserted values.
*
*    APPEND   The value of meaPoint is ignored,
*             the values are appended at the
*             end of the current values.
*
*    UPDATE   Update or modify the values at
*             meaPoint, the current values are
*             overwritten. If meaPoint is bigger
*             than the number of values in the
*             vector, the measurement point is
*             automatically appended.
*
* The names of the columns have to exist.
*
* When a client creates a ValueMatrix based on AoMeasurement
* this methods behaves as follows:
* The server checks the submatrices and creates all
* necessary instances of AoSubmatrix, AoLocalColumn and
* AoMeasurementQuantity. The values of the name attribute
* of AoSubmatrix must be generated by the server. The value
* will be equal to the value of the attribute ID (converted
* to DT_STRING). Missing instances of AoMeasurementQuantity
* will be created by the server, too.
*
* The mandatory attributes will get the following default
* values:
* Name supplied by client
* Datatype copied from AoQuantity.default_datatype
* Typesize copied from AoQuantity.default_typesize
* Interpolation no interpolation
* Rank copied from AoQuantity.default_rank
* Dimension copied from AoQuantity.default_dimension
*
* The server takes the value of the channel (supplied by
* client) and looks up the corresponding instance in
* AoQuantity using the attribute default_meq_name.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_IS_MEASUREMENT_MATRIX
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  set  The set type.
*
* @param  meaPoint  The measurement point.
*
* @param  value  The values to be inserted.
*/
void setValueMeaPoint(
   in SetType set,
   in T_LONG meaPoint,
   in NameValueSequence value)
   raises (AoException);

/* (22012)
* Create or modify a value vector in a value matrix.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The data is made permanent on transaction commit. Until
* the transaction is committed or until the access to the
* measurement point of the value matrix, it is allowed to
* have different number
* of values in the different value vectors.
*
* The meaning of the parameter setType is:
*
*    INSERT   Insert the values from startPoint,
*             the current available values from
*             startPoint are moved to the end of
*             the new inserted values.
*
*    APPEND   The value of startPoint is ignored,
*             the values are appended at the end
*             of the current values.
*
*    UPDATE   Update or modify the values from
*             startPoint, the current values are
*             overwritten. If the number of values
*             in the parameter values is too big
*             the values are automatically
*             appended.
*
* When a client creates a ValueMatrix based on AoMeasurement
* this methods behaves as follows:
* The server checks the submatrices and creates all
* necessary instances of AoSubmatrix, AoLocalColumn and
* AoMeasurementQuantity. The values of the name attribute
* of AoSubmatrix must be generated by the server. The value
* will be equal to the value of the attribute ID (converted
* to DT_STRING). Missing instances of AoMeasurementQuantity
* will be created by the server,
* too.
*
* The mandatory attributes will get the following default
* values:
* Name supplied by client
* Datatype copied from AoQuantity.default_datatype
* Typesize copied from AoQuantity.default_typesize
* Interpolation no interpolation
* Rank copied from AoQuantity.default_rank
* Dimension copied from AoQuantity.default_dimension
*
* The server takes the value of the channel (supplied by
* client) and looks up the corresponding instance in
* AoQuantity using the attribute default_meq_name.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_COLUMN
*    AO_INVALID_SET_TYPE
*    AO_IS_MEASUREMENT_MATRIX
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  col  The column whose values are to be set.
*
* @param  set  The set type.
*
* @param  startPoint  The starting point for the new values.
*
* @param  value  The values to be inserted.
*/
void setValueVector(
   in Column col,
   in SetType set,
   in T_LONG startPoint,
   in TS_ValueSeq value)
   raises (AoException);

/* (22013)
* Create or modify a number of value vectors in a value
* matrix.
*
* It is only allowed to create a value vector if the value
* matrix is created from a submatrix.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The data is made permanent on transaction commit. Until
* the transaction is committed or until the access to the
* measurement point of the value matrix it is allowed to
* have different numbers of values in the different value
* vectors.
*
* The names of the parameter values are the names of the
* columns. The values are the new values of the column.
* (setValueMeaPoint allows only one point, the TS_ValueSeq
* allows more then one point) There is a sequence of name
* value pairs (setValueVector allows only one column), so a
* block of values can be modified.
*
* The meaning of the parameter setType is:
*
*    INSERT   Insert the values from startPoint,
*             the current available values from
```

```
 *             startPoint are moved to the end of
 *             the new inserted values.
 *
 *    APPEND   The value of startPoint is ignored,
 *             the values are appended at the end
 *             of the current values.
 *
 *    UPDATE   Update or modify the values from
 *             startPoint, the current values are
 *             overwritten. If the number of values
 *             is greater than the number of values in the
 *             vector, the measurement point is
 *             automatically appended.
 *
 *    REMOVE   Remove the number of values from
 *             each column, starting with startPoint.
 *             The name of the column is given as the
 *             name of the NameValueseqUnit, the
 *             number of values to remove is given
 *             in the number of the values in the
 *             value.
 *
 * The names of the columns have to exist.
 *
 * When a client creates a ValueMatrix based on AoMeasurement
 * this methods behaves as follows:
 * The server checks the submatrices and creates all
 * necessary instances of AoSubmatrix, AoLocalColumn and
 * AoMeasurementQuantity. The values of the name attribute
 * of AoSubmatrix must be generated by the server. The value
 * will be equal to the value of the attribute ID (converted
 * to DT_STRING). Missing instances of AoMeasurementQuantity
 * will be created by the server, too.
 *
 * The mandatory attributes will get the following default
 * values:
 * Name supplied by client
 * Datatype copied from AoQuantity.default_datatype
 * Typesize copied from AoQuantity.default_typesize
 * Interpolation no interpolation
 * Rank copied from AoQuantity.default_rank
 * Dimension copied from AoQuantity.default_dimension
 *
 * The server takes the value of the channel (supplied by
 * client) and looks up the corresponding instance in
 * AoQuantity using the attribute default_meq_name.
```

```
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_COLUMN
*    AO_INVALID_SET_TYPE
*    AO_IS_MEASUREMENT_MATRIX
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  set  The set type.
*
* @param  startPoint  The measurement point.
*
* @param  value  The values to be inserted.
*/
void setValue(
   in SetType set,
   in T_LONG startPoint,
   in NameValueSeqUnitSequence value)
   raises (AoException);

/* (22014)
* Add a column to the value matrix. It is only allowed to
* create a value vector if the value matrix is created from
* a submatrix.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The data is made permanent on transaction commit. Until
* the transaction is committed or until the access to the
* measurement point of the value matrix it is allowed to
* have different number of values in the different value
* vectors. After the new column is added, it is possible to
* set the values of the column.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_NAME
*    AO_INVALID_SET_TYPE
*    AO_IS_MEASUREMENT_MATRIX
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  newColumn  The name and unit of the column to add.
*
* @return  The new column.
*/
Column addColumn(
   in NameUnit newColumn)
   raises (AoException);

/* (22015)
* Get the names of the scaling columns of the value matrix.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  colPattern  The name or the search pattern for the
*                     scaling column name.
*
* @return  The names of the scaling columns of the value
*          matrix.
*/
NameSequence listScalingColumns(
   in Pattern colPattern)
   raises (AoException);

/* (22016)
* Get the scaling column of the value matrix.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  colPattern  The name or the search pattern for the
*                     scaling column name.
*
* @return  The scaling columns of the value matrix.
*/
ColumnSequence getScalingColumns(
   in Pattern colPattern)
   raises (AoException);

/* (22017)
* List the names of the columns, which are scaled by the
* given column.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_NO_SCALING_COLUMN
*
* @param  scalingColumn  The scaling column.
*
* @return  The names of the columns which are scaled by the
*          given input column.
*/
NameSequence listColumnsScaledBy(
   in Column scalingColumn)
   raises (AoException);

/* (22018)
* Get the columns which are scaled by the given column.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_NO_SCALING_COLUMN
```

```
*
* @param  scalingColumn  The scaling column.
*
* @return  The columns which are scaled by the given input
*          column.
*/
ColumnSequence getColumnsScaledBy(
   in Column scalingColumn)
   raises (AoException);

/* (22019)
* Add a column to the value matrix. It is only allowed to
* create a value vector if the value matrix is created from
* a submatrix. The column will be scaled by the given
* scaling column.
*
* It is allowed to modify the object outside a transaction
* but it is recommended to activate a transaction.
*
* The data is made permanent on transaction commit. Until
* the transaction is committed or until the access to the
* measurement point of the value matrix it is allowed to
* have different number of values in the different value
* vectors. After the new column is added, it is possible to
* set the values of the column.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_INVALID_NAME
*    AO_INVALID_SET_TYPE
*    AO_IS_MEASUREMENT_MATRIX
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_NO_SCALING_COLUMN
*
* @param  newColumn  The name and unit of the column to add.
*
* @param  scalingColumn  The scaling column.
*
* @return  The new column.
*/
Column addColumnScaledBy(
   in NameUnit newColumn,
   in Column scalingColumn)
   raises (AoException);

/* (22020)
* Destroy the object on the server. The destructor of the
* client, so the server knows this object is not used
* anymore by the client. Access to this object after the
* destroy method will lead to an exception.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*/
void destroy()
   raises (AoException);

/* (22021)
* Get the values of different columns of the value matrix.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  columns  The requested columns.
*
* @param  startPoint  The starting point in the column.
*
* @param  count  The number of points to be retrieved. 0
*               mean until end of column.
*
* @return  The values of the different columns. The name of
*          the return structure corresponds with the name of
*          the column. The unit corresponds with the unit of
*          the column. The order of the resulst might not
```

```
   *          match the order in the requested sequence.
   */
   NameValueSeqUnitSequence getValue(
      in ColumnSequence columns,
      in T_LONG startPoint,
      in T_LONG count)
      raises (AoException);

}; // Interface ValueMatrix.

/*
* The ASAM ODS name-value-unitId iterator interface. This
* interface is identical with the NameValueUnitIterator, except
* the unit is given as an Id insead of a string.
*/
interface NameValueUnitIdIterator {

   /* (23001)
   * Destroy the iterator and free the associated memory.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   */
   void destroy()
      raises (AoException);

   /* (23002)
   * Get the total number of elements accessible by the
   * iterator.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The number of elements accessible by the
   *          iterator.
   */
```

```
T_LONG getCount()
   raises (AoException);

/* (23003)
* Get the next n elements from the sequence.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  how_many  The number of requested elements.
*
* @return  The next n name-value-unit tuples from the
*          name-value-unit tuple sequence.
*/
NameValueSeqUnitId nextN(
   in T_LONG how_many)
   raises (AoException);

/* (23004)
* Get the next element from the sequence.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The next name-value-unitId tuple from the
*          name-value-unitId tuple sequence.
*/
NameValueUnitId nextOne()
   raises (AoException);

/* (23005)
* Reset the pointer in the element sequence to the first
* element.
*
* @throws AoException
```

```
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   */
   void reset()
      raises (AoException);

}; // Interface NameValueUnitIdIterator.

/*
* The application element access.  The ApplElemAccess interface
* (Protocol level 3: GetVal, PutVal, GetInstRef, ...) is
* responsible to retrieve instances of application elements.
* Currently available query methods are:
* - query with attribute and value
* - query with attribute and values (not)in set
* - sorted results
* - grouped results including having condition
*/
interface ApplElemAccess {

   /* (24001)
   * Perform the Query.
   *
   * The number of different application elements which are
   * requested are exactly defined by the definition of the
   * query given in the field anuSeq of the QueryStructure. The
   * number of attributes for each application element is also
   * given in the definition. The number of matching instances
   * (their attributes) is not defined by the query and can be
   * a large amount. Therefore only one iterator for the
   * attribute values are defined.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  aoq  The query definition.
```

```
*
* @param  how_many  Maximum number of values in each result
*                   set. Valid arguments are:
*                   how_many = 0 : report all found values,
*                   how_many > 0 : report a maximum number
*                                  of values.
*
* @return  The result set with the requested attribute
*          values.
*/
ElemResultSetSequence getInstances(
   in QueryStructure aoq,
   in T_LONG how_many)
   raises (AoException);

/* (24002)
* Get related instances (Id). This method returns a sequence
* of related instances.
*
* The relation name specifies the relation given in the
* ApplStructValue. The aid of the ElemId and the relName
* defines the unique relation and the target application
* element.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  elem  Original instance. At the protocol level 3
*               this information was stored     in the field
*               elemId of the structure GetInstRefReq and
*               the request AOP_GetInstRef.
*
* @param  relName  The relation name. At the protocol level
*                  3 this information was stored in the
*                  field refName of the structure
*                  GetInstRefReq   and the request
*                  AOP_GetInstRef.
*
* @return  The list of the Id of the related instances.
*/
```

```
S_LONGLONG getRelInst(
   in ElemId elem,
   in Name relName)
   raises (AoException);

/* (24003)
* Set the instance reference.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  elem  The instance to add the related instances.
*               At the protocol level 3 this information was
*               stored in the field elemId1 of the structure
*               SetInstRefReq  and the request
*               AOP_SetInstRef.
*
* @param  relName  The name of relation. At the protocol
*                  level 3 this information was stored in
*                  the field refName of the structure
*                  SetInstRefReq and the request
*                  AOP_SetInstRef.
*
* @param  instIds  Sequence of instance id's. At the
*                  protocol level 3 this information was
*                  stored in the field elemId2 of the
*                  structure SetInstRefReq and the request
*                  AOP_SetInstRef. It was not possiable to
*                  set more then one relation.
*
* @param  type  The type of the modification, insert, update
*               or remove. At the protocol level 3 this
*               information was stored in the field onoff of
*               the structure SetInstRefReq and the request
*               AOP_SetInstRef.
*/
void setRelInst(
   in ElemId elem,
   in Name relName,
   in S_LONGLONG instIds,
   in SetType type)
   raises (AoException);

/* (24004)
* Create instance elements of an application element.
* The application element is specified by the AID of input
* structure.
* The attribute names are specified by the name of the input
* structure.
* The values of one instance element are specified in the
* valueseq of the input structure.
* You can create several instance elements in one call by
* filling the valueseq of the input structure.
* The same index in the valueseq corresponds to the
* attributes values of one instance element. This method
* returns a sequence of Id's, the order is related to the
* order of instance element specified in the input
* structure. In case of inheritance, the method supports
* only instances of same subtype per call. The returned Id's
* are the Id's of the related supertype instances.
*
* The client must supply all mandatory attributes and
* references within one single method call; otherwise the
* object cannot be made persistent by the server in the
* database without the risk of violating any database
* constraint.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  val  The sequence of attributes and their values.
*             At the protocol level 3 this information was
*             stored in the fields elemId and nvSeq of the
*             structure PutValReq and the request
*             AOP_PutValReq.
*
* @return  List with the Ids of the new created instances.
*/
```

```
ElemIdSequence insertInstances(
   in AIDNameValueSeqUnitIdSequence val)
   raises (AoException);

/* (24005)
* Update the one or more attributes of one or more instance
* elements. It is necessary that the input structure includes
* also the Id attribute, the Id attribute will be used to
* select the instance elements. In case of inherited
* application elements the supertype Id has to be included.
* The values of one instance element are specified in the
* valueseq of the input structure.
* The same index in the valueseq corresponds to the
* attributes values of one instance element.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  val  The sequence of attributes and their values.
*              At least one of the attribute values sequence
*              must be a sequence with the Id. At the
*              protocol level 3 this information was stored
*              in the fields elemId and nvSeq of the
*              structure PutValReq and the request
*              AOP_PutValReq.
*/
void updateInstances(
   in AIDNameValueSeqUnitIdSequence val)
   raises (AoException);

/* (24006)
* Delete instances from an application element. In case of
* inherited application elements the Id of the supertype has
* to be specified.
*
* This method can be used to delete several instances of the
* same application element, the method removeInstances
* remove one instance of an application element with the
* children of the instance element.
```

```
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  aid  The application element Id.
*
* @param  instIds  The sequence of instance Id's. At the
*                  protocol level 3 this information was
*                  stored in the fields elemId and nvSeq of
*                  the structure PutValReq and the request
*                  AOP_PutValReq.
*/
void deleteInstances(
   in T_LONGLONG aid,
   in S_LONGLONG instIds)
   raises (AoException);

/* (24007)
* Get the value matrix of a measurement or a submatrix. If
* the value matrix will be built up with special submatrix
* link, use the interface Measument.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_INVALID_BASETYPE
*
* @param  elem  The element id. The aid has to be the
*               appliction element Id of the measurment or
*               submatrix.
*
* @return  The value matrix
*/
ValueMatrix getValueMatrix(
   in ElemId elem)
   raises (AoException);

/* (24008)
* Set the ACL information on some application
* element-attribute defined by <aid> and <attr_name>. The
* <usergroup_id> defines the usergroup the rights should be
* set for. <rights> defines the rights to set or to clear.
* If the parameter <set> is set to 'set', the rights in
* <rights> are set, all others are cleared. If the parameter
* <set> is set to 'add', the rights in <rights> are added to
* the existing rights. If the parameter <set> is set to
* 'remove', the rights in <rights> are removed from the
* existing rights. Restriction for the model: only one
* application element of the type AoUserGroup is allowed.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  aid  The Id of the application element.
*
* @param  attrName  The name of the attribute.
*
* @param  usergroupId  The usergroup to set the rights for.
*
* @param  rights  The new right for the usergroup.
*
* @param  set  What to do with the new right.
*/
void setAttributeRights(
   in T_LONGLONG aid,
   in T_STRING attrName,
   in T_LONGLONG usergroupId,
   in T_LONG rights,
   in RightsSet set)
   raises (AoException);

/* (24009)
* Set the ACL information on some application element
* defined by <aid>. The <usergroup_id> defines the usergroup
* the rights should be set for. <rights> defines the rights
* to set or to clear. If the parameter <set> is set to
* 'set', the rights in <rights> are set, all others are
* cleared. If the parameter <set> is set to 'add', the
* rights in <rights> are added to the existing rights. If
* the parameter <set> is set to 'remove', the rights in
* <rights> are removed from the existing rights.
* Restriction for the model: only one application element of
* the type AoUserGroup is allowed.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  aid  The Id of the application element.
*
* @param  usergroupId  The usergroup to set the rights for.
*
* @param  rights  The new rights for the usergroup. The
*                 rights constants are defined in the
*                 interface SecurityRights. The interface
*                 definition language IDL does not allow to
*                 set the values of enumerations thus the
*                 constant definitions had to be done in an
*                 interface.
*
* @param  set  What to do with the new right.
*/
void setElementRights(
   in T_LONGLONG aid,
   in T_LONGLONG usergroupId,
   in T_LONG rights,
   in RightsSet set)
   raises (AoException);

/* (24010)
* Set the ACL information on some instance defined by the
* application element id <aid> and a sequence of instance
* defined by the id <iid>. The <usergroup_id> defines the
* usergroup the rights should be set for. <rights> defines
* the rights to set or to clear. If the parameter <set> is
* set to 'set', the rights in <rights> are set, all others
* are cleared. If the parameter <set> is set to 'add', the
* rights in <rights> are added to the existing rights. If
* the parameter <set> is set to 'remove', the rights in
* <rights> are removed from the existing rights.
* Restriction for the model: only one application element of
* the type AoUserGroup is allowed.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  aid  The Id of the application element.
*
* @param  instIds  The sequence of instance Id's.
*
* @param  usergroupId  The usergroup to set the rights for.
*
* @param  rights  The new right for the usergroup.
*
* @param  set  What to do with the new right.
*/
void setInstanceRights(
   in T_LONGLONG aid,
   in S_LONGLONG instIds,
   in T_LONGLONG usergroupId,
   in T_LONG rights,
   in RightsSet set)
   raises (AoException);

/* (24011)
* Retrieve access control list information for the given
* application attribute <aid>/<attr_name>.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aid  The Id of the application element.
*
* @param  attrName  The name of the attribute.
*
* @return  The access control list entries of the give
*          application attribute.
*/
ACLSequence getAttributeRights(
   in T_LONGLONG aid,
   in T_STRING attrName)
   raises (AoException);

/* (24012)
* Retrieve access control list information for the requested
* application element <aid>.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aid  The Id of the application element.
*
* @return  The access control list entries of the given
*          application element.
*/
ACLSequence getElementRights(
   in T_LONGLONG aid)
   raises (AoException);

/* (24013)
* Retrieve access control list information for the requested
* instance <aid>/<iid>.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aid  The Id of the application element.
*
* @param  iid  The Id of the instance.
*
* @return  The access control list entries of the given
*          instance.
*/
ACLSequence getInstanceRights(
   in T_LONGLONG aid,
   in T_LONGLONG iid)
   raises (AoException);

/* (24014)
* Set the access control list information for the initial
* rights on some application element defined by <aid>. The
* <usergroup_id> defines the usergroup the rights should be
* set for. <rights> defines the rights to set or to clear.
* If the parameter <set> is set to 'set', the rights in
* <rights> are set, all others are cleared. If the parameter
* <set> is set to 'add', the rights in <rights> are added to
* the existing rights. If the parameter <set> is set to
* 'remove', the rights in <rights> are removed from the
* existing rights.  Restriction for the model: only one
* application element of the type AoUserGroup is allowed.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  aid  The Id of the application element.
*
* @param  usergroupId  The usergroup to set the initial
*                      rights for.
*
```

```
* @param  rights  The new initial rights for the usergroup.
*                 The rights constants are defined in the
*                 interface SecurityRights. The interface
*                 definition language IDL does not allow to
*                 set the values of enumerations thus the
*                 constant definitions had to be done in an
*                 interface.
*
* @param  refAid  The Id of referencing application element
*                 for which the initial rights will be used.
*                 If no refAid is set the initial rights
*                 will be used for each new instance element
*                 independent of the application element.
*
* @param  set  What to do with the new initial rights.
*/
void setElementInitialRights(
   in T_LONGLONG aid,
   in T_LONGLONG usergroupId,
   in T_LONG rights,
   in T_LONGLONG refAid,
   in RightsSet set)
   raises (AoException);

/* (24015)
* Set the access control list information for the initial
* rights on some instance defined by the application element
* id <aid> and a sequence of instance defined by the id
* <iid>. The <usergroup_id> defines the usergroup the rights
* should be set for. <rights> defines the rights to set or
* to clear. If the parameter <set> is set to 'set', the
* rights in <rights> are set, all others are cleared. If the
* parameter <set> is set to 'add', the rights in <rights>
* are added to the existing rights. If the parameter <set>
* is set to 'remove', the rights in <rights> are removed
* from the existing rights.  Restriction for the model: only
* one application element of the type AoUserGroup is
* allowed.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  aid  The Id of the application element.
*
* @param  instIds  The sequence of instance Id's.
*
* @param  usergroupId  The usergroup to set the initial
*                      rights for.
*
* @param  rights  The new initial right for the usergroup.
*
* @param  refAid  The Id of referencing application element
*                 for which the initial rights will be used.
*                 If no refAid is set the initial rights
*                 will be used for each new instance element
*                 independent of the application element.
*
* @param  set  What to do with the new initial rights.
*/
void setInstanceInitialRights(
   in T_LONGLONG aid,
   in S_LONGLONG instIds,
   in T_LONGLONG usergroupId,
   in T_LONG rights,
   in T_LONGLONG refAid,
   in RightsSet set)
   raises (AoException);

/* (24016)
* Set the flag <set> in svcattr, if this reference will be
* used (or not) to retrieve the Initial Rights. If more than
* one reference is set to true, the union (or-function) of
* all rights are used.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @param  aid  The application element Id.
```

```
*
* @param  refName  The name of the reference.
*
* @param  set  What to do with the given reference.
*/
void setInitialRightReference(
   in T_LONGLONG aid,
   in T_STRING refName,
   in RightsSet set)
   raises (AoException);

/* (24017)
* Get all attribute names (references) which are used to
* retrieve the Initial Rights.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aid  The application element Id.
*
* @return  The names of the references which will be used
*          for the initial rights determination.
*/
NameSequence getInitialRightReference(
   in T_LONGLONG aid)
   raises (AoException);

/* (24018)
* Retrieve access control list information of the initial
* rights for the requested application element <aid>.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
```

```
*
* @param  aid  The Id of the application element.
*
* @return  The access control list entries of the given
*          application element for the initial rights.
*/
InitialRightSequence getElementInitialRights(
   in T_LONGLONG aid)
   raises (AoException);

/* (24019)
* Retrieve access control list information of the initial
* rights for the requested instance <aid>/<iid>.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  aid  The Id of the application element.
*
* @param  iid  The Id of the instance.
*
* @return  The access control list entries of the given
*          instance for the initial rights.
*/
InitialRightSequence getInstanceInitialRights(
   in T_LONGLONG aid,
   in T_LONGLONG iid)
   raises (AoException);

/* (24020)
* Perform the Query. This method can be used for a more
* powerful query compared to the method getInstances of
* this interface, with join definitions and aggregate
* functions.
*
* The number of different application elements which are
* requested are exactly defined by the definition of the
* query given in the field anuSeq of the QueryStructureExt.
* The number of attributes for each application element is
* also given in the definition. The number of matching
```

```
   * instances (their attributes) is not defined by the query
   * and can be a large amount. Therefore only one iterator for
   * the attribute values are defined.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  aoq  The query definition.
   *
   * @param  how_many  Maximum number of values in each result
   *                   set. Valid arguments are:
   *                   how_many = 0 : report all found values,
   *                   how_many > 0 : report a maximum number
   *                                  of values.
   *
   * @return  The result set with the requested attribute
   *          values.
   */
   ResultSetExtSequence getInstancesExt(
      in QueryStructureExt aoq,
      in T_LONG how_many)
      raises (AoException);

}; // Interface ApplElemAccess.

/*
* The ASAM ODS query evaluator interface allows to perform
* queries in synchronous mode and to create Query objects for
* asynchronous execution.
*/
interface QueryEvaluator {

   /* (25000)
   * Evaluate a query in synchronous mode.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_QUERY_TYPE_INVALID
   *    AO_QUERY_INVALID
   *    AO_QUERY_PROCESSING_ERROR
   *    AO_QUERY_TIMEOUT_EXCEEDED
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  queryStr  The query string.
   *
   * @param  params  Sequence of parameter names and values.
   *                 The following parameters should be
   *                 passed:
   *
   *                 Name: "MaxDuration";
   *                 Type: T_LONG
   *                 Comment: Can be used to restrict the
   *                 processing time. The time is given in
   *                 milliseconds,
   *                 Default value: 0 (no restriction)
   *
   * @return  The result of the query as an instance element
   *          iterator.
   */
   InstanceElementIterator getInstances(
      in T_STRING queryStr,
      in NameValueSequence params)
      raises (AoException);

   /* (25001)
   * Evaluate a query in synchronous mode.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_QUERY_TYPE_INVALID
   *    AO_QUERY_INVALID
   *    AO_QUERY_PROCESSING_ERROR
   *    AO_QUERY_TIMEOUT_EXCEEDED
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @param  queryStr  The query string.
```

```
*
* @param  params  Sequence of parameter names and values.
*                 The following parameters should be
*                 passed:
*
*                 Name: "MaxDuration";
*                 Type: T_LONG
*                 Comment: Can be used to restrict the
*                 processing time. The time is given in
*                 milliseconds,
*                 Default value: 0 (no restriction)
*
* @return  The result of the query as a name value unit
*          sequence iterator. Each name value unit tuple is
*          one cell of the table. The name value unit
*          sequence is one row of the table.
*/
NameValueUnitSequenceIterator getTableRows(
   in T_STRING queryStr,
   in NameValueSequence params)
   raises (AoException);

/* (25002)
* Evaluate a query in synchronous mode.
*
* @throws AoException
* with the following possible error codes:
*    AO_QUERY_TYPE_INVALID
*    AO_QUERY_INVALID
*    AO_QUERY_PROCESSING_ERROR
*    AO_QUERY_TIMEOUT_EXCEEDED
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  queryStr  The query string.
*
* @param  params  Sequence of parameter names and values.
*                 The following parameters should be
*                 passed:
*
*                 Name: "MaxDuration";
*                 Type: T_LONG
*                 Comment: Can be used to restrict the
*                 processing time. The time is given in
*                 milliseconds,
*                 Default value: 0 (no restriction)
*
* @return  The result of the query as a name value sequence
*          unit sequence. Each name value sequecne unit
*          tuple is one column of the table. The name value
*          sequence unit sequence is  the table. The result
*          structure can be very huge.
*/
NameValueSeqUnitSequence getTable(
   in T_STRING queryStr,
   in NameValueSequence params)
   raises (AoException);

/* (25003)
* Create a query object to execute it in asynchronous mode.
*
* @throws AoException
* with the following possible error codes:
*    AO_QUERY_TYPE_INVALID
*    AO_QUERY_INVALID
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  queryStr  The query string
*
* @param  params  Sequence of parameter names and values.
*                 The following parameters should be
*                 passed:
*
*                 Name: "QueryResultType";
*                 Type: ResultType.
*                 Comment: Specifies what kind of result is
*                 expected by the client.
*                 Default value:
*                 INSTELEM_ITERATOR_AS_RESULT
*
*                 Name: "MaxDuration";
*                 Type: T_LONG
*                 Comment: Can be used to restrict the
```

```
   *                  processing time. The time is given in
   *                  milliseconds,
   *                  Default value: 0 (no restriction)
   *
   * @return  The query object.
   */
   Query createQuery(
      in T_STRING queryStr,
      in NameValueSequence params)
      raises (AoException);

}; // Interface QueryEvaluator.

/*
* The ASAM ODS interface Query.
*/
interface Query {

   /* (26000)
   * Get the QueryEvaluator object which is responsible for
   * this query.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *    AO_TRANSACTION_NOT_ACTIVE
   *
   * @return  The QueryEvaluator object which is responsible
   *          for this query.
   */
   QueryEvaluator getQueryEvaluator()
      raises (AoException);

   /* (26001)
   * Do the query pre-processing (optimization, etc.)  Call can
   * be omited by the client. In this case the functionality
   * is executed on the call of executeQuery.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *    AO_QUERY_PROCESSING_ERROR
   *    AO_QUERY_INVALID_RESULTTYPE
   *
   * @param  params  Sequence of parameter names and values.
   *                 The following parameters should be
   *                 passed:
   *
   *                 Name: "QueryResultType";
   *                 Type: ResultType.
   *                 Comment: Specifies what kind of result is
   *                 expected by the client, this parameter is
   *                 required if the parameters isn't given at
   *                 the method createQuery of the interface
   *                 QueryEvaluator.
   *                 Default value: INSTELEM_ITERATOR_AS_RESULT
   */
   void prepareQuery(
      in NameValueSequence params)
      raises (AoException);

   /* (26002)
   * Execute query in asynchronous mode.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *    AO_QUERY_PROCESSING_ERROR
   *    AO_QUERY_INVALID_RESULTTYPE
   *
   * @param  params  Sequence of parameter names and values.The
   *                 following parameter should be passed:
   *
   *                 Name: "QueryResultType";
   *                 Type: ResultType.
   *                 Comment: Specifies what kind of result is
   *                 expected by the client, this parameter is
   *                  required if the parameters isn't given at
   *                  the method prepareQuery or the method
   *                  createQuery of the interface
   *                  QueryEvaluator.
   *                  Default value:
   *                  INSTELEM_ITERATOR_AS_RESULT
   *
   *                  Name: "Synchronous";
   *                  Type: T_BOOLEAN
   *                  Comment: In case of "true" guarantees
   *                  synchronous execution.
   *                  Default value: "false"
   */
   void executeQuery(
      in NameValueSequence params)
      raises (AoException);

   /* (26003)
   * Return query status.
   *
   * Returns INCOMPLETE if the query is still executing.
   *
   * Returns COMPLETE if the query finished execution or if the
   * query execution stopped because of an error or because the
   * timeout was exceeded.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The status of the query.
   */
   QueryStatus getStatus()
      raises (AoException);

   /* (26004)
   * Get the query result. This method should only be called
   * after the query has been executed. It returns an iterator
   * on the instances that were found by the query.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *    AO_QUERY_PROCESSING_ERROR
   *    AO_QUERY_TIMEOUT_EXCEEDED
   *    AO_QUERY_INCOMPLETE
   *    AO_QUERY_INVALID_RESULTTYPE
   *
   * @return  The result of the query as an instance element
   *          iterator.
   */
   InstanceElementIterator getInstances()
      raises (AoException);

   /* (26005)
   * Get the query result. This method should only be called
   * after the query has been executed. It returns an iterator
   * on the instances that were found by the query.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *    AO_QUERY_PROCESSING_ERROR
   *    AO_QUERY_TIMEOUT_EXCEEDED
   *    AO_QUERY_INCOMPLETE
   *    AO_QUERY_INVALID_RESULTTYPE
   *
   * @return  The result of the query as a name value unit
   *          sequence iterator. Each name value unit tuple is
   *          one cell of the table. The name value unit
   *          sequence is one row of the table.
   */
   NameValueUnitSequenceIterator getTableRows()
      raises (AoException);

   /* (26006)
   * Get the query result. This method should only be called
```

```
   * after the query has been executed. It returns an iterator
   * on the instances that were found by the query.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *    AO_QUERY_PROCESSING_ERROR
   *    AO_QUERY_TIMEOUT_EXCEEDED
   *    AO_QUERY_INCOMPLETE
   *    AO_QUERY_INVALID_RESULTTYPE
   *
   * @return  The result of the query as a name value sequence
   *          unit sequence. Each name value sequecne unit
   *          tuple is one column of the table. The name value
   *          sequence unit sequence is  the table. The result
   *          structure can be very huge.
   */
   NameValueSeqUnitSequence getTable()
      raises (AoException);

}; // Interface Query.

/*
* The name value unit sequence iterator. The table iterator as
* query result.
*/
interface NameValueUnitSequenceIterator {

   /* (27001)
   * Destroy the iterator and free the associated memory.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   */
   void destroy()
      raises (AoException);
```

```
/* (27002)
* Get the total number of elements accessible by the
* iterator.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The number of elements accessible by the
*          iterator.
*/
T_LONG getCount()
   raises (AoException);

/* (27003)
* Get the next n elements from the sequence.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  how_many  The number of requested elements.
*
* @return  The next n values of the name-value-unit
*          sequence. For each NameValuUnit the next n values
*          are stored in the value sequence.
*/
NameValueSeqUnitSequence nextN(
   in T_LONG how_many)
   raises (AoException);

/* (27004)
* Get the next element from the iterator.
*
* @throws AoException
* with the following possible error codes:
      *    AO_CONNECTION_LOST
      *    AO_IMPLEMENTATION_PROBLEM
      *    AO_NOT_IMPLEMENTED
      *    AO_NO_MEMORY
      *    AO_SESSION_NOT_ACTIVE
      *
      * @return  The next name-value-unit tuple sequence from the
      *          name-value-unit tuple.
      */
      NameValueSeqUnit nextOne()
         raises (AoException);

      /* (27005)
      * Reset the pointer in the element iterator to the first
      * element.
      *
      * @throws AoException
      * with the following possible error codes:
      *    AO_CONNECTION_LOST
      *    AO_IMPLEMENTATION_PROBLEM
      *    AO_NOT_IMPLEMENTED
      *    AO_NO_MEMORY
      *    AO_SESSION_NOT_ACTIVE
      */
      void reset()
         raises (AoException);

   }; // Interface NameValueUnitSequenceIterator.

   /*
   * The ASAM ODS enumeration interface.
   */
   interface EnumerationDefinition {

      /* (28000)
      * List the possible names of the enumeration. The sort order
      * of the list is the value of the item. The first item has
      * the value zero.
      *
      * @throws AoException
      * with the following possible error codes:
      *    AO_BAD_PARAMETER
      *    AO_CONNECTION_LOST
      *    AO_IMPLEMENTATION_PROBLEM
      *    AO_NOT_IMPLEMENTED
      *    AO_NO_MEMORY
```

```
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @return  List with all possiable names of the enumeration
*          items.
*/
NameSequence listItemNames()
   raises (AoException);

/* (28001)
* Get the value of an item.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*    AO_NOT_FOUND
*
* @param  itemName  The name of the item.
*
* @return  The number of the item.
*/
T_LONG getItem(
   in T_STRING itemName)
   raises (AoException);

/* (28002)
* Get the name of an item.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*    AO_NOT_FOUND
*
* @param  item  The value of the item.
```

```
*
* @return  The name of the item.
*/
T_STRING getItemName(
   in T_LONG item)
   raises (AoException);

/* (28003)
* Add a new item to the enumeration. This method modifies
* the application model and is only allowed for the
* superuser.
*
* The name of an item must not exceed the maximum name
* length of the underlying physical storage. The current
* physical storage specification restricts it to 128 characters.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*    AO_ACCESS_DENIED
*    AO_DUPLICATE_NAME
*    AO_DUPLICATE_VALUE
*
* @param  itemName  The name of the new item.
*/
void addItem(
   in T_STRING itemName)
   raises (AoException);

/* (28004)
* Rename the item of the enumeration. This method modifies
* the application model and is only allowed for the
* superuser.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*    AO_NOT_FOUND
*
* @param  oldItemName  The existing name of the itrem.
*
* @param  newItemName  the new name of the item.
*/
void renameItem(
   in T_STRING oldItemName,
   in T_STRING newItemName)
   raises (AoException);

/* (28005)
* Get the name of the enumeration.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*    AO_TRANSACTION_NOT_ACTIVE
*
* @return  Name of the enumeration.
*/
T_STRING getName()
   raises (AoException);

/* (28006)
* Set the name of the enumeration. This method modifies the
* application model and is only allowed for the superuser.
*
* The name of an enumeration definition must not exceed the
* maximum name length of the underlying physical storage.
* The current physical storage specification restricts it to 30
* characters.
*
* @throws AoException
* with the following possible error codes:
*    AO_BAD_PARAMETER
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
```

```
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *    AO_TRANSACTION_NOT_ACTIVE
   *    AO_ACCESS_DENIED
   *
   * @param  enumName  Name of the enumeration.
   */
   void setName(
      in T_STRING enumName)
      raises (AoException);

   /* (28007)
   * Get the index of the enumeration.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_BAD_PARAMETER
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *    AO_TRANSACTION_NOT_ACTIVE
   *
   * @return  The index of the enumeration.
   */
   T_LONG getIndex()
      raises (AoException);

}; // Interface EnumerationDefinition.

/*
* For iteration through the result elements, there is also a
* new iterator necessary called ElemResultSetExtSeqIterator.
* It's functionality is still the same as all other iterators
* and needs therefore no further explanation.
* There is only one difference to the other iterations. The
* iteration is done on the lowest level in that case on the
* TS_UnionSeq.
*/
interface ElemResultSetExtSeqIterator {

   /* (29000)
   * Destroy the iterator and free the associated memory.
   *
```

```
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*/
void destroy()
   raises (AoException);

/* (29001)
* Get the total number of elements accessible by the
* iterator.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @return  The number of elements accessible by the
*          iterator.
*/
T_LONG getCount()
   raises (AoException);

/* (29002)
* Get the next n elements from the sequence.
*
* @throws AoException
* with the following possible error codes:
*    AO_CONNECTION_LOST
*    AO_IMPLEMENTATION_PROBLEM
*    AO_NOT_IMPLEMENTED
*    AO_NO_MEMORY
*    AO_SESSION_NOT_ACTIVE
*
* @param  how_many  The number of requested elements.
*
* @return  The next n attribute values from the element
*          result set sequence.
*/
ElemResultSetExtSequence nextN(
      in T_LONG how_many)
      raises (AoException);

   /* (29003)
   * Get the next  element from the sequence.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   *
   * @return  The next attribute values from the element result
   *          set sequence.
   */
   ElemResultSetExt nextOne()
      raises (AoException);

   /* (29004)
   * Reset the pointer in the element sequence to the first
   * element.
   *
   * @throws AoException
   * with the following possible error codes:
   *    AO_CONNECTION_LOST
   *    AO_IMPLEMENTATION_PROBLEM
   *    AO_NOT_IMPLEMENTED
   *    AO_NO_MEMORY
   *    AO_SESSION_NOT_ACTIVE
   */
   void reset()
      raises (AoException);

}; // Interface ElemResultSetExtSeqIterator.

}; // Module ods.

}; // Module asam.

}; // Module org.


#endif
```

```
/*
* The ASAM ODS error codes.
*
* AO_ACCESS_DENIED
* The remote server denied the access. If this error occurred
* it was not even possible to present the authentication
* information. This means the authentication information might
* be correct but the server refused the access already at a
* lower level.
*
* AO_BAD_OPERATION
* The BAD_OPERATION error code is used when a method is invalid
* in a marshalling operation.
*
* AO_BAD_PARAMETER
* A parameter of the wrong type was passed to the method. The
* minorCode tells which parameter (1, 2, 3 or 4) is bad. If
* more than
* one parameter is bad, only the first one is reported. This
* error can occur only in non-typesave language bindings. Those
* language bindings also impose the problem that not all
* parameter errors are automatically detectable.
*
* AO_CONNECTION_LOST
* Due to a hardware or network software problem the connection
* to the server was lost.
*
* AO_CONNECT_FAILED
* The connect to a server failed. This error may occur if the
* server is down or currently unreachable.
*
* AO_CONNECT_REFUSED
* The connection was refused by the server. This error may
* occur if the presented authentication information is either
* incorrect or incomplete. This error shall not occur if the
* server does not accept any more sessions due to overload
* problems. See AO_SESSION_LIMIT_REACHED for this case.
*
* AO_DUPLICATE_BASE_ATTRIBUTE
* Any application element may have only one base attribute of a
* certain type. This means it may have only one attribute of
* base attribute type NAME, one ID, one VERSION and so on.
*
* AO_DUPLICATE_NAME
* The implicit or explicit specified name is already in use but
* it is required to be unique.
```

```
*
* AO_DUPLICATE_VALUE
* The attribute is marked unique in the application model. Thus
* duplicate values are not allowed.
*
* AO_HAS_BASE_ATTRIBUTE
* Base attribute found. It is not allowed to modify the
* datatype, unique- or obligatory flag .
*
* AO_HAS_BASE_RELATION
* Base relation found. It is not allowed to modify the
* relationtype, -range or -ship of an application relation
* derived from a base relation.
*
* AO_HAS_INSTANCES
* The operation is not allowed for elements that have
* instances.
*
* AO_HAS_REFERENCES
* The requested operation is not permitted because the target
* element has references.
*
* AO_IMPLEMENTATION_PROBLEM
* This error is reserved for the reporting of implementation
* specific problems that are not properly handled by the
* standard error definitions. An application should not crash
* if this error occurs but there is no way to react to this
* error other than reporting and ignoring
* it. The intend of this error is not to leave implementation
* specific errors unreported.
*
* AO_INCOMPATIBLE_UNITS
* The units are incompatible. No conversion rule is known.
*
* AO_INVALID_ASAM_PATH
* The specified Asam path is invalid.
*
* AO_INVALID_ATTRIBUTE_TYPE
* The requested attribute type is invalid.
*
* AO_INVALID_BASETYPE
* The specified base type is invalid. The following basetypes
* are allowed:
*    AoAny
*    AoAttributeMap
*    AoEnvironment
*    AoLocalColumn
*    AoLog
*    AoMeasurement
*    AoMeasurementQuantity
*    AoNameMap
*    AoParameter
*    AoParameterSet
*    AoPhysicalDimension
*    AoQuantity
*    AoQuantityGroup
*    AoSubmatrix
*    AoSubTest
*    AoTest
*    AoTestDevice
*    AoTestEquipment
*    AoTestEquipmentPart
*    AoTestSequence
*    AoTestSequencePart
*    AoUnit
*    AoUnitGroup
*    AoUnitUnderTest
*    AoUnitUnderTestPart
*    AoUser
*    AoUserGroup
*
* AO_INVALID_BASE_ELEMENT
* The base element is invalid in this context. If this is an
* element of type measurement, another element of this type may
* already exist.
*
* AO_INVALID_BUILDUP_FUNCTION
* The specified build-up function is invalid.
*
* AO_INVALID_COLUMN
* The specified column is invalid.
*
* AO_INVALID_COUNT
* The specified number of points is invalid (probably
* negative).
*
* AO_INVALID_DATATYPE
* The datatype is not allowed in the given context or it
* conflicts with an existing datatype definition.
*
```

```
* This error may also occur in non-typesave language bindings.
* To avoid this error in all language bindings it is
* recommended to use always the definitions of the enumeration
* "DataType".
*
* AO_INVALID_ELEMENT
* The element is invalid in this context.
*
* AO_INVALID_LENGTH
* The given length is invalid. Negative length values are not
* allowed.
*
* AO_INVALID_ORDINALNUMBER
* The ordinal number is either already used or less than zero.
*
* AO_INVALID_RELATION
* The relation is invalid. The related elements and the base
* relation do not fit.
*
* AO_INVALID_RELATIONSHIP
* This error may occur only in non-typesave language bindings.
* To avoid this error in all language bindings it is
* recommended to use always the definitions of the enumeration
* "Relationship".
*
* AO_INVALID_RELATION_RANGE
* The specified relation range is invalid.
*
* AO_INVALID_RELATION_TYPE
* This error may occur only in non-typesave language bindings.
* To avoid this error in all language bindings it is
* recommended to use always the definitions of the enumeration
* "RelationType".
*
* AO_INVALID_SET_TYPE
* The specified set-type is invalid.
*
* AO_INVALID_SMATLINK
* The submatrix link is invalid. Either submatrix 1 or 2 is not
* specified or the ordinal number is missing when there is more
* than one SMatLink.
*
* AO_INVALID_SUBMATRIX
* The specified submatrix is invalid.
*
* AO_IS_BASE_ATTRIBUTE
* The application attribute is already of a base attribute
* type. It can not be changed. If this is required, the
* application attribute has to be removed from its application
* element and re-created. This error may occur if an
* application attribute derived from a base attribute
*
*    a. shall be overwritten by another base
*       attribute type.
*    b. shall receive another datatype.
*    c. shall receive another unique-flag.
*    d. shall receive another obligatory-flag.
*
* AO_IS_BASE_RELATION
* Properties of base relations may not be changed.
*
* AO_IS_MEASUREMENT_MATRIX
* The matrix is a complex, generated matrix from a measurement
* not just a simple submatrix. It is only allowed to modify
* submatrices but not the composed measurement matrices.
*
* AO_MATH_ERROR
* A computation error occurred. This can be an overflow, an
* underflow or a division by zero.
*
* AO_MISSING_APPLICATION_ELEMENT
* A required application element is missing.
*
* AO_MISSING_ATTRIBUTE
* A required (obligatory) attribute is missing.
*
* AO_MISSING_RELATION
* A required relation is missing.
*
* AO_MISSING_VALUE
* An obligatory value is missing (the AO_VF_DEFINED flag is
* zero).
*
* AO_NOT_FOUND
* The requested element was not found. This error occurs only
* in remove or rename operations if the subject of the
* operation is not found. All get- and list-methods return an
* empty list if the requested item is not found.
*
* AO_NOT_IMPLEMENTED
* The requested method is not yet implemented. This error is
* not allowed to occur in a certified implementation. It is
* intended to allow partial operational tests. during the
* development process.
*
* AO_NOT_UNIQUE
* This error occurs if the instances of a property are required
* to be unique.
*
* AO_NO_MEMORY
* No more volatile memory available.
*
* AO_NO_PATH_TO_ELEMENT
* A free-floating element was detected. No navigation path
* leads to this element.
*
* AO_NO_SCALING_COLUMN
* The column is no scaling column.
*
* AO_OPEN_MODE_NOT_SUPPORTED
* The requested open mode is not supported. Valid open modes
* are "read" and "write". Anything else is rejected with this
* error and no session is created.
*
* AO_QUERY_INCOMPLETE
* The execution of the query was not yet completed.
*
* AO_QUERY_INVALID
* Some error in the query string or some inconsistency between
* the return type of the query string and the  result type
* specified by parameter "QueryResultType".
*
* AO_QUERY_INVALID_RESULTTYPE
* The requested result type of the query do no metch with the
* previous definition of the result type.
*
* AO_QUERY_PROCESSING_ERROR
* Some error occured during the execution of the query.
*
* AO_QUERY_TIMEOUT_EXCEEDED
* It was not possible to execute the query within the  time
* limit set by parameter "MaxDuration".
*
* AO_QUERY_TYPE_INVALID
* The server does not support the specified query language
* type.
*
* AO_SESSION_LIMIT_REACHED
* The server does not accept any new connections. This error
* may occur if the server reached the session limit for a
* distinct user or the total number of sessions allowed.
*
* AO_SESSION_NOT_ACTIVE
* The session is no longer active. This error occurs if an
* attempt is made to call a method of a closed session. This
* error shall not be confused with the error
* AO_CONNECTION_LOST.
*
* AO_TRANSACTION_ALREADY_ACTIVE
* There may be only one active transaction at one time. If this
* error occurs there is already an active transaction. That
* transaction remains active in case of this error.
*
* AO_TRANSACTION_NOT_ACTIVE
* Write operation have to be done always in the context of a
* transaction. This error occurs if no transaction is active
* during a write operation.
*
* AO_UNKNOWN_ERROR
* Use the zero as unknown error to avoid confusing error
* messages if no error code has been set.
*
* AO_UNKNOWN_UNIT
* The unit is unknown.
*/
```

10.6 USING ASAM HARMONIZED DATATYPES

ASAM ODS has defined data types in a very early state of specification. They are preceded by T_ resp. S_. Those data types have been used in a variety of implementations. In the meantime ASAM has harmonized the data types between all ASAM standards. Section 2.5 describes the relationship between ASAM-wide harmonized data types and the legacy data types of ASAM ODS. Since there are several implementations available basing on the ASAM ODS data types, it was decided to keep them in the OO-API. However, anyone who intends to base on the harmonized data types according to section 2.5 may do so.

This section provides an IDL-file which contains the specification of the harmonized data types.

It should be noted that the ASAM ODS DataType enumeration is part of the standardized ASAM data types; the enumeration item names (starting with DT_ resp. DS_) are officially released by ASAM for all working groups and may be used without any restriction.

10.6.1 MAPPING FILE A_TYPES.IDL

For mapping the harmonized data types, the following file a_types.idl is provided. This is for informational purposes only; the actual version is maintained independently from ASAM ODS and may be requested from ASAM e.V..

```
// *****************************************************************
// ***  ASAM Data Type Specification                             ***
// ***  File    : a_types.idl                                    ***
// ***  Version : 1.1                                            ***
// ***  Date    : January 20, 2003                               ***
// *****************************************************************
//

//--------------- Start of the a_types.IDL file ---------------------

// base types

typedef boolean           A_BOOLEAN;           // boolean
typedef char              A_INT8;              // 8 bit integer
typedef octet             A_UINT8;             // 8 bit unsigned integer
typedef short             A_INT16;             // 16 bit integer
typedef unsigned short    A_UINT16;            // 16 bit unsigned integer
typedef long              A_INT32;             // 32 bit integer
typedef unsigned long     A_UINT32;            // 32 bit unsigned integer
typedef float             A_FLOAT32;           // 32 bit float
typedef double            A_FLOAT64;           // 64 bit float

typedef A_UINT8           A_BCD;               // BCD number

typedef A_INT32           A_ENUM;              // Enumeration

//---------------------------------------------------------------

// structures describing numeric values
// 64 bit integer
struct A_INT64 {
   A_INT32       n32High;
   A_UINT32      un32Low;
};

// 64 bit unsigned integer
struct A_UINT64 {
   A_UINT32      un32High;
   A_UINT32      un32Low;
};

// complex value (real and imaginary part are float values)
struct A_COMPLEX32 {
   A_FLOAT32     fReal;
   A_FLOAT32     fImaginary;
};

// complex value (real and imaginary part are double values)
struct A_COMPLEX64 {
   A_FLOAT64     dReal;
   A_FLOAT64     dImaginary;
};
//------------------------------------------------------------

// character types

typedef A_UINT8  A_ASCII;      // semantically, one char 1..255
                               // or the termination character Hex00
typedef A_UINT16 A_UNICODE_2;  // semantically: one 16 bit character
defined in
                               // ISO-10646 UCS-2 0x0001 - 0xFFFF
                               // or the termination character Hex0000
typedef A_INT32  A_UNICODE_4;  // semantically: one 31 bit character
defined in
                               // ISO-10646 UCS-4  0x00000001 - 0x7FFFFFFF
                               // or the termination character Hex00000000

//------------------------------------------------------------

// string types

// ASCII string

typedef sequence <A_ASCII>       A_ASCIISTRING;    // max restricted to
4294967295
typedef sequence <A_ASCII, 64>   A_ASCIISTRING_64;
typedef sequence <A_ASCII, 128>  A_ASCIISTRING_128;
typedef sequence <A_ASCII, 255>  A_ASCIISTRING_255;
typedef sequence <A_ASCII, 1024> A_ASCIISTRING_1024;
typedef sequence <A_ASCII, 2048> A_ASCIISTRING_2048;
typedef sequence <A_ASCII, 5100> A_ASCIISTRING_5100;
typedef sequence <A_ASCII, 2>    A_ASCIISTRING_2;

// UCS-2 string

typedef sequence <A_UNICODE_2>   A_UNICODE2STRING; // max restricted to
4294967295
```

```
                                                     // zero terminated with
                                                     // character Hex0000
// UCS-4 string

typedef sequence <A_UNICODE_4>   A_UNICODE4STRING; // max restricted to
4294967295
                                                     // zero terminated with
                                                     // character Hex00000000

// character field types

// ASCII character field
struct A_ASCIIFIELD {
   A_UINT32      un32Chars;   // value range un32Chars: [1, 2^32-1]
   A_ASCIISTRING szValue;     // un32Chars contains the number of ASCII
                              // characters in the character field
};

// UCS-2 character field
struct A_UNICODE2FIELD {
   A_UINT32         un32Chars; // value range un32Chars: [1, 2^32-1]
   A_UNICODE2STRING szValue;   // un32Chars contains the number of UCS-2
                               // characters in the character field
};


// UCS-4 character field
struct A_UNICODE4FIELD {
   A_UINT32         un32Chars; // value range un32Chars: [1, 2^32-1]
   A_UNICODE4STRING szValue;   // un32Chars contains the number of UCS-4
                               // characters in the character field
};
//----------------------------------------------------------------

// bit and byte fields

typedef sequence <A_UINT8, 256> BYTEARRAY_256;

struct A_BITFIELD {
   A_INT16          n16NumberOfBits; // number of valid bits in bitfield
                                     // [1 .. 2048]
   BYTEARRAY_256    aun8Field;       // array of bytes,
                                     // length of un8Field is calculated as
                                     // (un32NumberOfBits + 7)/8
                                     // value range for number of bytes:
[1..256]
};

typedef sequence <A_UINT8> BYTEARRAY;  // maximum restricted to 4294967295

struct ASAM_BYTEFIELD {
   A_UINT32  un32Bytes;             // value range un32Bytes:[1, 2^32-1]
   BYTEARRAY aun8Field;             // array of bytes
};
//----------------------------------------------------------------

// IP-address (IPv4)
```

```
struct A_IP4 {
   A_UINT8 un8Octet1;
   A_UINT8 un8Octet2;
   A_UINT8 un8Octet3;
   A_UINT8 un8Octet4;
};


// IP-address (IPv6)

struct A_IP6 {
   A_UINT16 un16Value1;
   A_UINT16 un16Value2;
   A_UINT16 un16Value3;
   A_UINT16 un16Value4;
   A_UINT16 un16Value5;
   A_UINT16 un16Value6;
   A_UINT16 un16Value7;
   A_UINT16 un16Value8;
};
//---------------------------------------------------------------

// data types for time instants

struct A_TIME_STRUCT {
   A_INT16  n16Year;           // year
   A_INT8   n8Month;           // month
   A_INT8   n8DayOfMonth;      // day of month
   A_INT8   n8Hour;            // hour
   A_INT8   n8Minute;          // minute
   A_INT8   n8Second;          // second
   A_INT16  n16MilliSecond;    // millisecond
   A_INT16  n16MicroSecond;    // microsecond
   A_INT16  n16NanoSecond;     // nanosecond
   A_INT32  n32TimeZoneDiff;   // difference between UTC and
                               // local time in seconds
};


// description of day within a year (year, month, day of month)

struct A_Day_YMD {
   A_INT16  n16Year;           // year
   A_INT8   n8Month;           // month
   A_INT8   n8DayOfMonth;      // day of month
};


// description of day within a year (year, week, day of week)

struct A_Day_YWD {
   A_INT16  n16Year;           // year
   A_INT8   n8Week;            // week
   A_INT8   n8DayOfWeek;       // day of week
};


// description of day within a year (year, day of year)
```

```
struct A_Day_YD {
   A_INT16  n16Year;          // year
   A_INT16  n16DayOfYear;     // day of year
};


//--------------------------------------------------------------

// Definition of types used inside URL adress


// Host access type is switch type of union
enum HOST_ACCESS_TYPE {
   SERVER_IP4,              // access via IP4 address
   SERVER_IP6,           // access via IP6 address
   SERVER_DOMAIN         // access via domain name
};

// structure describing host access using IPv4 address
struct HOST_IP4 {
   A_ASCIISTRING szUser;         // user
   A_ASCIISTRING szPwd;                // password
   A_IP4         strAddress;     // IP4 address
   A_INT32       n32Port;        // port number
};

// structure describing host access using IPv6 address
struct HOST_IP6 {
   A_ASCIISTRING szUser;         // user
   A_ASCIISTRING szPwd;                // password
   A_IP6        strAddress;      // IP6 address
   A_INT32      n32Port;         // port number
};

// domain name
typedef A_ASCIISTRING_5100 A_DOMAIN_NAME;

// structure describing host access using domain name
struct HOST_DOMAIN {
   A_ASCIISTRING szUser;         // user
   A_ASCIISTRING szPwd;                // password
   A_DOMAIN_NAME szDomain;       // domain name
   A_INT32       n32Port;        // port number
};


// union describing host access
union HOST_ACCESS switch (HOST_ACCESS_TYPE) {
   case SERVER_IP4 :   HOST_IP4        hostIP4;
   case SERVER_IP6 :   HOST_IP6        hostIP6;
   default         :   HOST_DOMAIN     hostDomain;
};


// URL address

struct A_URL {
   A_ASCIISTRING    AccessScheme;      // e.g. ftp, http,..
   HOST_ACCESS_TYPE HostAccessType;
   HOST_ACCESS      HostAccess;
   A_ASCIISTRING    szPath;                  // path at the server
};
//--------------------------------------------------------------
// definition of country and language

typedef A_ASCIISTRING_2 A_COUNTRY;
typedef A_ASCIISTRING_2 A_LANGUAGE;



// datatype enumerations for persistent storages.
// do not insert new names - only append is allowed
// never append new types without ASAM ODS permission
// because of possible ambiguities in data archives
enum DataType {
   DT_UNKNOWN,            // Unknown datatype.
   DT_STRING,             // String.
   DT_SHORT,              // A_INT16
   DT_FLOAT,              // A_FLOAT32
   DT_BOOLEAN,            // A_BOOLEAN
   DT_BYTE,               // A_INT8
   DT_LONG,               // A_INT32
   DT_DOUBLE,             // A_FLOAT64
   DT_LONGLONG,           // A_INT64
   DT_ID,                 // LongLong value (64 bit). Not used. DT_LONGLONG
is used instead.
   DT_DATE,               // Date.
   DT_BYTESTR,            // Bytestream.
   DT_BLOB,               // Blob.
   DT_COMPLEX,            // A_COMPLEX32
   DT_DCOMPLEX,           // A_COMPLEX64
   DS_STRING,             // String sequence.
   DS_SHORT,              // Short sequence.
   DS_FLOAT,              // Float sequence.
   DS_BOOLEAN,            // Boolean sequene.
   DS_BYTE,               // Byte sequence.
   DS_LONG,               // Long sequence.
   DS_DOUBLE,             // Double sequence.
   DS_LONGLONG,           // Longlong sequence.
   DS_COMPLEX,            // Complex sequence.
   DS_DCOMPLEX,           // Double complex sequence.
   DS_ID,                 // LongLong sequence. Not used. DS_LONGLONG is
used instead.
   DS_DATE,               // Date sequence.
   DS_BYTESTR,            // Bytestream sequence.
   DT_EXTERNALREFERENCE,  // External reference.
   DS_EXTERNALREFERENCE   // Sequence of external reference.
};


//---------------- End of the a_types.IDL file ----------------------
```

10.7 QUICKREFERENCE

This chapter contains a quick reference for the most important OO-API Interfaces and their respective methods. CORBA IDL syntax is used for a language independent representation of return values and parameters. Detailed descriptions and examples can be found in the respective language binding of the OO-API (see section 10.2).

Note: This chapter is intended for expert users only. Novice users might find it helpful to get a comprehensive overview (see section 10.2).

10.7.1 INTERFACE AOFACTORY

- `T_STRING` **`getDescription`**`()`
- `T_STRING` **`getInterfaceVersion`**`()`
- `T_STRING` **`getName`**`()`
- `T_STRING` **`getType`**`()`
- `AoSession` **`newSession`**`(T_STRING auth)`

10.7.2 INTERFACE AOSESSION

- `void a`**`bortTransaction`**`()`
- `void` **`close`**`()`
- `void` **`commitTransaction`**`()`
- `Blob` **`createBlob`**`()`
- `QueryEvaluator` **`createQueryEvaluator`**`()`
- `void` **`flush`**`()`
- `ApplElemAccess` **`getApplElemAccess`**`()`
- `ApplicationStructure` **`getApplicationStructure`**`()`
- `ApplicationStructureValue` **`getApplicationStructureValue`**`()`
- `BaseStructure` **`getBaseStructure`**`()`
- `NameValueIterator` **`getContext`**`(Pattern varPattern)`
- `NameValue` **`getContextByName`**`(Name varName)`
- `T_STRING` **`getDescription`**`()`
- `LockMode` **`getLockMode`**`()`
- `Name` **`getName`**`()`
- `T_STRING` **`getType`**`()`
- `NameIterator` **`listContext`**`(Pattern varPattern)`
- `void` **`removeContext`**`(Pattern varPattern)`
- `void` **`setContext`**`(NameValue contextVariable)`
- `void` **`setContextString`**`(Name varName, T_STRING value)`
- `void` **`setCurrentInitialRights`**`(InitialRightSequence irlEntries, T_BOOLEAN set)`
- `void` **`setLockMode`**`(LockMode lockMode)`
- `void` **`setPassword`**`(T_STRING username, T_STRING oldPassword, T_STRING newPassword)`
- `void` **`startTransaction`**`()`

10.7.3 INTERFACE APPLELEMACCESS

- ➢ void **deleteInstances**(T_LONGLONG aid, S_LONGLONG instIds)
- ➢ ACLSequence **getAttributeRights**(T_LONGLONG aid, T_STRING attrName)
- ➢ InitialRightSequence **getElementInitialRights**(T_LONGLONG aid)
- ➢ ACLSequence **getElementRights**(T_LONGLONG aid)
- ➢ NameSequence **getInitialRightReference**(T_LONGLONG aid)
- ➢ InitialRightSequence **getInstanceInitialRights**(T_LONGLONG aid, T_LONGLONG instId)
- ➢ ACLSequence **getInstanceRights**(T_LONGLONG aid, T_LONGLONG instId)
- ➢ ElemResultSetSequence **getInstances**(QueryStructure aoq, T_LONG how_many)
- ➢ ResultSetExtSequence **getInstancesExt**(QueryStructureExt aoq, T_LONG how_many)
- ➢ S_LONGLONG **getRelInst**(ElemId elem, Name relName)
- ➢ ValueMatrix **getValueMatrix**(ElemId elem)
- ➢ ElemIdSequence **insertInstances**(AIDNameValueSeqUnitIdSequence val)
- ➢ void **setAttributeRights**(T_LONGLONG aid, T_STRING attrName, T_LONGLONG usergroupId, T_LONG rights, RightsSet set)
- ➢ void **setElementInitialRights**(T_LONGLONG aid, T_LONGLONG usergroupId, T_LONG rights, T_LONGLONG refAid, RightsSet set)
- ➢ void **setElementRights**(T_LONGLONG aid, T_LONGLONG usergroupId, T_LONG rights, RightsSet set)
- ➢ void **setInitialRightReference**(T_LONGLONG aid, T_STRING refName, RightsSet set)
- ➢ void **setInstanceInitialRights**(T_LONGLONG aid, S_LONGLONG instIds, T_LONGLONG usergroupId, T_LONG rights, T_LONGLONG refAid, RightsSet set)
- ➢ void **setInstanceRights**(T_LONGLONG aid, S_LONGLONG instIds, T_LONGLONG usergroupId, T_LONG rights, RightsSet set)
- ➢ void **setRelInst**(ElemId elem, Name relName, S_LONGLONG instIds, SetType type)
- ➢ void **updateInstances**(AIDNameValueSeqUnitIdSequence val)

10.7.4 INTERFACE APPLICATIONATTRIBUTE

- ➢ ApplicationElement **getApplicationElement**()
- ➢ BaseAttribute **getBaseAttribute**()
- ➢ DataType **getDataType**()
- ➢ EnumerationDefinition **getEnumerationDefinition**()
- ➢ T_LONG **getLength**()
- ➢ Name **getName**()
- ➢ ACLSequence **getRights**()
- ➢ T_LONGLONG **getUnit**()
- ➢ T_BOOLEAN **hasUnit**()
- ➢ T_BOOLEAN **hasValueFlag**()

- `T_BOOLEAN` **`isAutogenerated`**`()`
- `T_BOOLEAN` **`isObligatory`**`()`
- `T_BOOLEAN` **`isUnique`**`()`
- `void` **`setBaseAttribute`**`(BaseAttribute baseAttr)`
- `void` **`setDataType`**`(DataType dataType)`
- `void` **`setEnumerationDefinition`**`(EnumerationDefinition enumDef)`
- `vois` **`setIsAutogenerated`**`(T_BOOLEAN isAutogenerated)`
- `void` **`setIsObligatory`**`(T_BOOLEAN isObligatory)`
- `void` **`setIsUnique`**`(T_BOOLEAN isUnique)`
- `void` **`setLength`**`(T_LONG length)`
- `void` **`setName`**`(Name name)`
- `void` **`setRights`**`(InstanceElement usergroup, T_LONG rights, RightsSet set)`
- `void` **`setUnit`**`(T_LONGLONG unitId)`
- `void` **`withUnit`**`(T_BOOLEAN withUnit)`
- `void` **`withValueFlag`**`(T_BOOLEAN withValueFlag)`

10.7.5 INTERFACE APPLICATIONELEMENT

- `ApplicationAttribute` **`createAttribute`**`()`
- `InstanceElement` **`createInstance`**`(Name ieName)`
- `InstanceElementSequence` **`createInstances`**`(NameValueSeqUnitSequence attributes, ApplicationRelationInstanceElementSeqSequence relatedInstances)`
- `ApplicationElementSequence` **`getAllRelatedElements`**`()`
- `ApplicationRelationSequence` **`getAllRelations`**`()`
- `ApplicationStructure` **`getApplicationStructure`**`()`
- `ApplicationAttribute` **`getAttributeByBaseName`**`(Name baName)`
- `ApplicationAttribute` **`getAttributeByName`**`(Name aaName)`
- `ApplicationAttributeSequence` **`getAttributes`**`(Pattern aaPattern)`
- `BaseElement` **`getBaseElement`**`()`
- `T_LONGLONG` **`getId`**`()`
- `ApplicationRelationSequence` **`getInitialRightRelations`**`()`
- `InitialRightSequence` **`getInitialRights`**`()`
- `InstanceElement` **`getInstanceById`**`(T_LONGLONG ieId)`
- `InstanceElement` **`getInstanceByName`**`(Name ieName)`
- `InstanceElementIterator` **`getInstances`**`(Pattern iePattern)`
- `Name` **`getName`**`()`
- `ApplicationElementSequence` **`getRelatedElementsByRelationship`**`(Relationship aeRelationship)`
- `ApplicationRelationSequence` **`getRelationsByType`**`(RelationType aeRelationType)`
- `ACLSequence` **`getRights`**`()`
- `SecurityLevel` **`getSecurityLevel`**`()`
- `NameSequence` **`listAllRelatedElements`**`()`
- `NameSequence` **`listAttributes`**`(Pattern aaPattern)`

- `NameIterator` **`listInstances`**`(Pattern aaPattern,`
- `NameSequence` **`listRelatedElementsByRelationship`**`(Relationship aaRelationship)`
- `void` **`removeAttribute`**`(ApplicationAttribute applAttr)`
- `void` **`removeInstance`**`(T_LONGLONG ieId, T_BOOLEAN recursive)`
- `void` **`setBaseElement`**`(BaseElement baseElem)`
- `void` **`setInitialRightRelation`**`(ApplicationRelation applRel, T_BOOLEAN set)`
- `void` **`setInitialRights`**`(InstanceElement usergroup, T_LONG rights, T_LONGLONG refAid, RightsSet set)`
- `void` **`setName`**`(Name aeName)`
- `void` **`setRights`**`(InstanceElement usergroup, T_LONG rights, RightsSet set)`
- `void` **`setSecurityLevel`**`(T_LONG secLevel, RightsSet set)`

10.7.6 INTERFACE APPLICATIONRELATION

- `BaseRelation` **`getBaseRelation`**`()`
- `ApplicationElement` **`getElem1`**`()`
- `ApplicationElement` **`getElem2`**`()`
- `Name` **`getInverseRelationName`**`()`
- `RelationRange` **`getInverseRelationRange`**`()`
- `Relationship` **`getInverseRelationship`**`()`
- `Name` **`getRelationName`**`()`
- `RelationRange` **`getRelationRange`**`()`
- `Relationship` **`getRelationship`**`()`
- `RelationType` **`getRelationType`**`()`
- `void` **`setBaseRelation`**`(BaseRelation baseRel)`
- `void` **`setElem1`**`(ApplicationElement applElem)`
- `void` **`setElem2`**`(ApplicationElement applElem)`
- `void` **`setInverseRelationName`**`(Name arInvName)`
- `void` **`setInverseRelationRange`**`(RelationRange arRelationRange)`
- `void` **`setRelationName`**`(Name arName)`
- `void` **`setRelationRange`**`(RelationRange arRelationRange)`
- `void` **`setRelationType`**`(RelationType arRelationType)`

10.7.7 INTERFACE APPLICATIONSTRUCTURE

- `void` **`check`**`()`
- `ApplicationElement` **`createElement`**`(BaseElement baseElem)`
- `EnumerationDefinition` **`createEnumerationDefinition`**`(T_STRING enumName)`
- `void` **`createInstanceRelations`**`(ApplicationRelation applRel, InstanceElementSequence elemList1, InstanceElementSequence elemList2)`
- `ApplicationRelation` **`createRelation`**`()`
- `ApplicationElement` **`getElementById`**`(T_LONGLONG aeId)`
- `ApplicationElement` **`getElementByName`**`(Name aeName)`

- ApplicationElementSequence **getElements**(Pattern aePattern)
- ApplicationElementSequence **getElementsByBaseType**(BaseType aeType)
- EnumerationDefinition **getEnumerationDefinition**(T_STRING enumName)
- InstanceElement **getInstanceByAsamPath**(Name asamPath)
- InstanceElementSequence **getInstancesById**(ElemIdSequence ieIds)
- ApplicationRelationSequence **getRelations**(ApplicationElement applElem1, ApplicationElement applElem2)
- AoSession **getSession**()
- ApplicationElementSequence **getTopLevelElements**(BaseType aeType)
- NameSequence **listElements**(Pattern aePattern)
- NameSequence **listElementsByBaseType**(BaseType aeType)
- NameSequence **listEnumerations**()
- NameSequence **listTopLevelElements**(BaseType aeType)
- void **removeElement**(ApplicationElement applElem)
- void **removeEnumerationDefinition**(T_STRING enumName)
- void **removeRelation**(ApplicationRelation applRel)

10.7.8 INTERFACE BASEATTRIBUTE

- BaseElement **getBaseElement**()
- DataType **getDataType**()
- Name **getName**()
- T_BOOLEAN **isObligatory**()
- T_BOOLEAN **isUnique**()

10.7.9 INTERFACE BASEELEMENT

- BaseRelationSequence **getAllRelations**()
- BaseAttributeSequence **getAttributes**(Pattern baPattern)
- BaseElementSequence **getRelatedElementsByRelationship**(Relationship brRelationship)
- BaseRelationSequence **getRelationsByType**(RelationType brRelationType)
- BaseType **getType**()
- T_BOOLEAN **isTopLevel**()
- NameSequence **listAttributes**(Pattern baPattern)
- BaseTypeSequence **listRelatedElementsByRelationship**(Relationship brRelationship)

10.7.10 INTERFACE BASERELATION

- BaseElement **getElem1**()
- BaseElement **getElem2**()
- RelationRange **getInverseRelationRange**()
- Relationship **getInverseRelationship**()
- Name **getRelationName**()

- RelationRange **getRelationRange**()
- Relationship **getRelationship**()
- RelationType **getRelationType**()

10.7.11 INTERFACE BASESTRUCTURE

- BaseElement **getElementByType**(BaseType beType)
- BaseElementSequence **getElements**(Pattern bePattern)
- BaseRelation **getRelation**(BaseElement elem1, BaseElement elem2)
- BaseElementSequence **getTopLevelElements**(Pattern bePattern)
- T_STRING **getVersion**()
- BaseTypeSequence **listElements**(Pattern bePattern)
- BaseTypeSequence **listTopLevelElements**(Pattern bePattern)

10.7.12 INTERFACE BLOB

- void **append**(S_BYTE value)
- T_BOOLEAN **compare**(T_BLOB aBlob)
- void **destroy**()
- S_BYTE **get**(T_LONG offset, T_LONG length)
- T_STRING **getHeader**()
- T_LONG **getLength**()
- void **set**(S_BYTE value)
- void **setHeader**(T_STRING header)

10.7.13 INTERFACE COLUMN

- void **destroy**()
- DataType **getDataType**()
- T_STRING **getFormula**()
- Name **getName**()
- InstanceElement **getSourceMQ**()
- T_STRING **getUnit**()
- T_BOOLEAN **isIndependent**()
- T_BOOLEAN **isScaling**()
- void **setFormula**(T_STRING formula)
- void **setIndependent**(T_BOOLEAN independent)
- void **setScaling**(T_BOOLEAN scaling)
- void **setUnit**(T_STRING unit)

10.7.14 INTERFACE ELEMRESULTSETEXTSEQITERATOR

- void **destroy**()
- T_LONG **getCount**();

- `ElemResultSetExtSequence` **`nextN`**`(T_LONG how_many)`
- `ElemResultSetExt` **`nextOne`**`()`
- `void` **`reset`**`()`

10.7.15 INTERFACE ENUMERATIONDEFINITION

- `void` **`addItem`**`(T_STRING itemName)`
- `T_LONG` **`getIndex`**`()`
- `T_LONG` **`getItem`**`(T_STRING itemName)`
- `T_STRING` **`getItemName`**`(T_LONG item)`
- `T_STRING` **`getName`**`()`
- `NameSequence` **`listItemNames`**`()`
- `void` **`renameItem`**`(T_STRING oldItemName, T_STRING newItemName)`
- `void` **`setName`**`(T_STRING enumName)`

10.7.16 INTERFACE INSTANCEELEMENT

- `void` **`addInstanceAttribute`**`(NameValueUnit instAttr)`
- `T_LONGLONG` **`compare`**`(InstanceElement compIeObj)`
- `InstanceElementSequence` **`createRelatedInstances`**`(ApplicationRelation applRel, NameValueSeqUnitSequence attributes, ApplicationRelationInstanceElementSeqSequence relatedInstances)`
- `void` **`createRelation`**`(ApplicationRelation relation, InstanceElement instElem)`
- `InstanceElement` **`deepCopy`**`(T_STRING newName, T_STRING newVersion))`
- `void` **`destroy`**`()`
- `ApplicationElement` **`getApplicationElement`**`()`
- `Name` **`getAsamPath`**`()`
- `T_LONGLONG` **`getId`**`()`
- `InitialRightSequence` **`getInitialRights`**`()`
- `Name` **`getName`**`()`
- `InstanceElementIterator` **`getRelatedInstances`**`(ApplicationRelation applRel, Pattern iePattern)`
- `InstanceElementIterator` **`getRelatedInstancesByRelationship`**`( Relationship ieRelationship, Pattern iePattern)`
- `ACLSequence` **`getRights`**`()`
- `NameValueUnit` **`getValue`**`(Name attrName)`
- `NameValueUnit` **`getValueByBaseName`**`(Name baseAttrName)`
- `NameValueUnit` **`getValueInUnit`**`(NameUnit attr)`
- `NameValueUnitSequence` **`getValueSeq`**`(NameSequence attrNames)`
- `NameSequence` **`listAttributes`**`(Pattern iaPattern, AttrType aType)`
- `NameIterator` **`listRelatedInstances`**`(ApplicationRelation ieRelation, Pattern iePattern)`
- `NameIterator` **`listRelatedInstancesByRelationship`**`(in  Relationship ieRelationship, Pattern, iePattern)`

- void **removeInstanceAttribute**(Name attrName)
- void **removeRelation**(ApplicationRelation applRel, InstanceElement instElem_nm)
- void **renameInstanceAttribute**(Name oldName, Nme newName)
- void **setInitialRights**(InstanceElement usergroup, T_LONG rights, T_LONGLONG refAid, RightsSet set)
- void **setName**(Name iaName)
- void **setRights**(InstanceElement usergroup, T_LONG rights, RightsSet set)
- void **setValue**(NameValueUnit value)
- void **setValueSeq**(NameValueUnitSequence values)
- InstanceElement **shallowCopy**(T_STRING newName, T_STRING newVersion)
- Measurement **upcastMeasurement**()
- SubMatrix **upcastSubMatrix**()

10.7.17 INTERFACE INSTANCEELEMENTITERATOR

- void **destroy**()
- T_LONG **getCount**();
- InstanceElementSequence **nextN**(T_LONG how_many)
- InstanceElement **nextOne**()
- void **reset**()

10.7.18 INTERFACE MEASUREMENT

- SMatLink **createSMatLink**()
- SMatLinkSequence **getSMatLinks**()
- ValueMatrix **getValueMatrix**()
- void **removeSMatLink**(SMatLink smLink)

10.7.19 INTERFACE NAMEITERATOR

- void **destroy**()
- T_LONG **getCount**();
- NameSequence **nextN**(T_LONG how_many)
- Name **nextOne**()
- void **reset**()

10.7.20 INTERFACE NAMEVALUEITERATOR

- void **destroy**()
- T_LONG **getCount**();
- NameValueSequence **nextN**(T_LONG how_many)
- NameValue **nextOne**()
- void **reset**()

10.7.21 INTERFACE NAMEVALUEUNITIDITERATOR

- void **destroy**()
- T_LONG **getCount**();
- NameValueUnitIdSequence **nextN**(T_LONG how_many)
- NameValueUnitId **nextOne**()
- void **reset**()

10.7.22 INTERFACE NAMEVALUEUNITITERATOR

- void **destroy**()
- T_LONG **getCount**();
- NameValueUnitSequence **nextN**(T_LONG how_many)
- NameValueUnit **nextOne**()
- void **reset**()

10.7.23 INTERFACE NAMEVALUEUNITSEQUENCEITERATOR

- void **destroy**()
- T_LONG **getCount**();
- NameValueSeqUnitSequence **nextN**(T_LONG how_many)
- NameValueSeqUnit **nextOne**()
- void **reset**()

10.7.24 INTERFACE QUERY

- void **executeQuery**(NameValueSequence params)
- InstanceElementIterator **getInstances**()
- QueryEvaluator **getQueryEvaluator**()
- QueryStatus **getStatus**()
- NameValueSeqUnitSequence **getTable**()
- NameValueUnitSequenceIterator **getTableRows**()
- void **prepareQuery**(NameValueSequence params)

10.7.25 INTERFACE QUERY

- Query **createQuery**(T_STRING queryStr, NameValueSequence params)
- InstanceElementIterator **getInstances**(T_STRING queryStr, NameValueSequence params)
- NameValueSeqUnitSequence **getTable**(T_STRING queryStr, NameValueSequence params)
- NameValueUnitSequenceIterator **getTableRows**(T_STRING queryStr, NameValueSequence params)

10.7.26 INTERFACE SMATLINK

- BuildUpFunction **getLinkType**()
- T_LONG **getOrdinalNumber**()
- SubMatrix **getSMat1**()
- ColumnSequence **getSMat1Columns**()
- SubMatrix **getSMat2**()
- ColumnSequence **getSMat2Columns**()
- void **setLinkType**(BuildUpFunction linkType)
- void **setOrdinalNumber**(T_LONG ordinalNumber)
- void **setSMat1**(SubMatrix subMat1)
- void **setSMat1Columns**(ColumnSequence columns)
- void **setSMat2**(SubMatrix subMat2)
- void **setSMat2Columns**(in ColumnSequence columns)

10.7.27 INTERFACE SUBMATRIX

- ColumnSequence **getColumns**(Pattern colPattern)
- ValueMatrix **getValueMatrix**()
- NameSequence **listColumns**(Pattern colPattern)

10.7.28 INTERFACE VALUEMATRIX

- Column **addColumn**(NameUnit newColumn)
- Column **addColumnScaledBy**(NameUnit newColumn, Column scalingColumn)
- void **destroy**()
- T_LONG **getColumnCount**()
- ColumnSequence **getColumns**(Pattern colPattern)
- ColumnSequence **getColumnsScaledBy**(Column scalingColumn)
- ColumnSequence **getIndependentColumns**(Pattern colPattern)
- T_LONG **getRowCount**()
- ColumnSequence **getScalingColumns**(Pattern colPattern)
- NameValueSeqUnitSequence **getValue**(ColumnSequence columns, T_LONG startPoint, T_LONG count)
- NameValueUnitIterator **getValueMeaPoint**(T_LONG meaPoint,
- TS_ValueSeq **getValueVector**(Column col, T_LONG startPoint, T_LONG count)
- NameSequence **listColumns**(Pattern colPattern)
- NameSequence **listColumnsScaledBy**(Column scalingColumn)
- NameSequence **listIndependentColumns**(Pattern colPattern)
- NameSequence **listScalingColumns**(Pattern colPattern)
- void **removeValueMeaPoint**(NameSequence columnNames, T_LONG meaPoint, T_LONG count)
- void **removeValueVector**(Column col, T_LONG startPoint, T_LONG count)
- void **setValue**(SetType set, T_LONG startPoint, NameValueSeqUnitSequence value)

- `void` **`setValueMeaPoint`**`(SetType set, T_LONG meaPoint, NameValueSequence value)`
- `void` **`setValueVector`**`(Column col, SetType set, T_LONG startPoint, TS_ValueSeq value)`

10.8 REVISION HISTORY

Date Editor	Changes
2004-03-04 R. Bartz	compared content of this chapter with ODS-IDL of version rc3 (Dec. 2003); updated 10.2, 10.5, and 10.7 accordingly

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 11
NVH APPLICATION MODEL

Version 1.3

Association for **S**tandardization of
Automation and **M**easuring Systems

Dated: 30.09.2004

Status of Document

Reference:	ASAM ODS Version 5.0 NVH Application Model
Date:	30.09.2004
Author:	Gert Sablon, LMS; Karsten Rucker, Müller-BBM; Dr. Wartini, Müller-BBM; Elmar Klinkenberg, Head Acoustics
Type:	Specification
Doc-ID:	ASAM_ODS_50_CH11_NVH_Model.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 (odsfrt@asam.net) to make sure this issue will be addressed within the current review cycle.

Contents

11 NVH APPLICATION MODEL 3

11.1 INTRODUCTION 3
11.2 HOW TO DESCRIBE THE NVH PARAMETERS 5
11.3 TYPES OF NVH DATA 5
11.4 1D DATA 6
11.5 2D DATA 7
11.5.1 GENERAL DESCRIPTION 7
11.5.2 STREAMED 9
11.5.3 REFERENCED STREAMED 10
11.5.4 SPECTRUM 11
11.5.5 REFERENCED SPECTRUM 12
11.5.6 OCTAVE SPECTRUM 13
11.5.7 CUT 13
11.5.8 CHOICES DEFINITIONS 14
11.5.9 PHYSICAL STORAGE 17
11.6 3D DATA 21
11.6.1 GENERAL DESCRIPTION 21
11.6.2 PHYSICAL STORAGE 22
11.7 THE MTL DATA TYPE 26
11.7.1 GENERAL DESCRIPTION 26
11.7.2 PHYSICAL STORAGE 28
11.8 THE RAINFLOW DATA TYPE 30
11.8.1 GENERAL DESCRIPTION 30
11.8.2 PHYSICAL STORAGE 32
11.9 THE TAL DATA TYPE 35
11.9.1 GENERAL DESCRIPTION 35
11.9.2 PHYSICAL STORAGE 37
11.10 QUANTITIES AND UNITS 39
11.10.1 QUANTITIES 39
11.10.2 UNITS 40
11.11 REVISION HISTORY 43

Scope

This document is a brief description of the application model for NVH (noise, vibration, and harshness) data in ASAM ODS Version 5.0.

Intended Audience

This document is intended for people interested in ASAM ODS Version 5.0 and the NVH application model. It shall be used a reference that describes this specific application model.

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

11 NVH APPLICATION MODEL

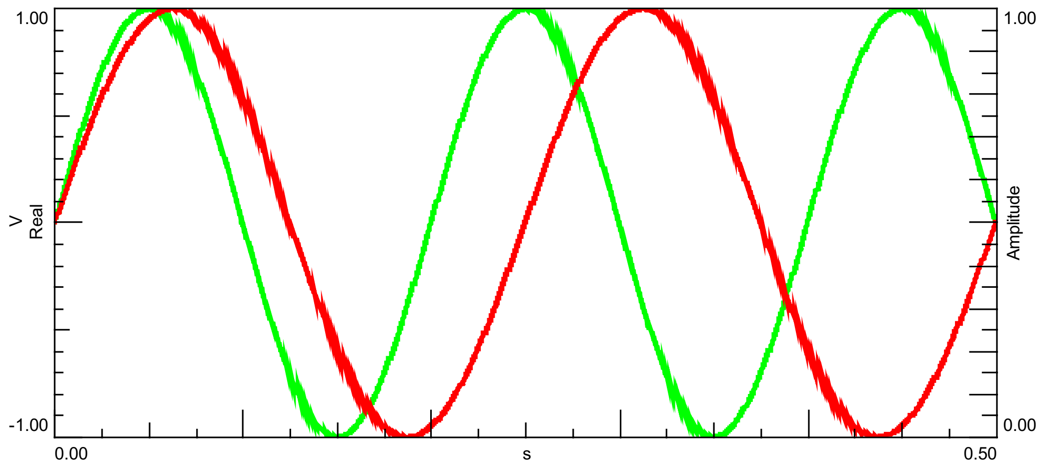
11.1 INTRODUCTION

This chapter describes the application model for NVH Data (NVH Data Model) as proposed by the corresponding ASAM ODS workgroup. The NVH Data Model is based on the ASAM ODS base model, and it is intended to be a minimal application model. This means that anyone who sets up an application model, and wants his database to include NVH data, needs at least all application elements proposed in this document.

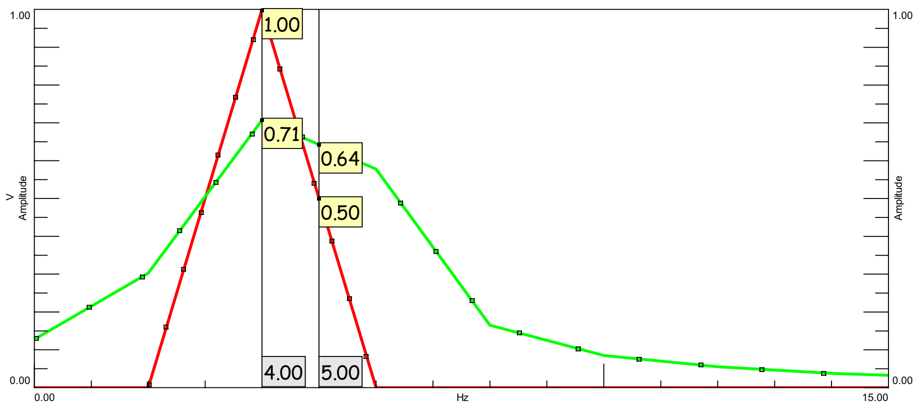
NVH stands for Noise, Vibration and Harshness, and it is a specific field of activity in the industry, where a lot of simulation results and measurement data exist. Since there exist quite a lot of software solutions for problem solving in the NVH field, being able to share data between all these solutions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

This can best be explained by means of an example. This example starts from 2 sine waves, both of them measured during 0,5 s. The first sine wave has a frequency of 4 Hz (red), the second one 5 Hz (green). Both sine waves have an amplitude of 1 V.

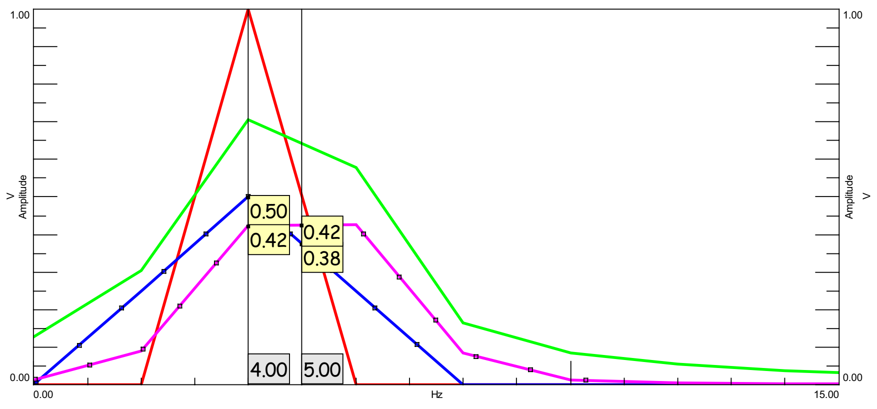


A typical operation is the Fast Fourier Transform, by means of which we can transfer both sine waves to the frequency domain. Now, if we do this for both sine waves, we get the following result.

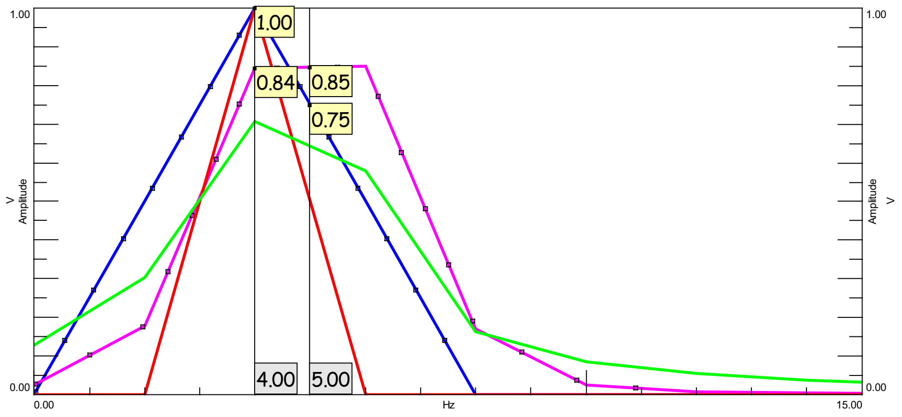


The sine wave with frequency 4 Hz nicely shows up as a peak at 4 Hz with amplitude of 1 V, but the sine wave with frequency of 5 Hz does not look so nice. This is due to the fact that this sine wave was not measured in the right way. The Discrete Fourier Transform will only show the right result when the signal is measured periodically, meaning that an integer number of periods of the sine wave must be measured to give the correct result in the frequency domain. The 4 Hz sine wave has exactly 2 periods in the 0,5 s measurement, but the 5 Hz sine wave has 2,5 periods, which is not an integer number. This results is what is called leakage: the energy of the sine wave at 5 Hz leaks away to the neighboring frequencies, resulting in a broader peak with a too low amplitude.

When measuring physical signals on cars or airplanes, it can never be guaranteed that the measurement is nicely periodic for all information in the signal, a technique to better condition this problem is needed, and digital signal processing provides such a technique: windowing. By applying a window to the measured time domain signal first, the non-periodicity of it is attenuated, but the signal itself too, so the result after windowing contains less energy, and thus will show up with a smaller amplitude in the frequency domain.



But knowing the window shape, this reduction in amplitude or energy can be compensated for by multiplying the result with what is called the window correction factor.



As can be seen on the last figure, after correction with the window correction factor, the sine wave at 4 Hz again gets an amplitude of 1 V – which is correct – and the sine wave at 5 Hz looks better now: it is easier to see that the center of the peak is at 5 Hz, and the amplitude is 0,85 V instead of the 0,64 V before windowing.

Of course, this is only possible if the window that was used and the corresponding window correction factor is known. This information must be stored separately, but together with the data, because further signal processing operation might require undoing the window correction or changing it into something else. Therefore, this kind of information is typically stored in attributes that go along with the data.

11.2 HOW TO DESCRIBE THE NVH PARAMETERS

Because it was not the intention to change existing ASAM ODS application models by adding all the necessary application attributes to correctly describe NVH data, the workgroup decided to make use of the AoParameterSet and AoParameter elements for storing this extra information.

Some few applic.attributes needed to be introduced (LocalColumn)

Names of Appl.El. of AoPar...

Usage of Attributes of AoPar..

Names of instance elements are specified... ==Table header “Parameter name”

11.3 TYPES OF NVH DATA

There are a lot of different data types in NVH data. The easiest way to divide them in categories is by using the dimensionality of the data.

Using that approach, this proposal will contain descriptions on how to store the following types of data:

- 1D (one-dimensional)
- 2D (two-dimensional)
- 3D (three-dimensional)

In the remainder of this document, each of these data types will be explained in more detail, and a table will be given containing all parameters needed to describe the corresponding data type. The column in this table, titled "Flag" denotes whether this parameter is really necessary for correct interpretation of this data type. This means that the real minimal application model should only include instances of type AoParameter for these parameters where this last column contains the value "required".

All data types for the parameters are specified by means of the data type enumeration name (see chapter 2.5 for more details on data types and their enumerations).

11.4 1D DATA

This data type is just a sequence of numbers, or a vector in other words. Therefore, it is called 1D data. This data type is typically used to store e.g. tacho pulse information. It will be described using the following parameters.

Parameter name	**DataType enum name**	**Flag**
creation time	DT_DATE	optional
last modification time	DT_DATE	optional
pulses per rev	DT_FLOAT	required
acquisition sample rate	DT_FLOAT	optional
carrier frequency	DT_FLOAT	required

MIME Types for 1D Data	**MIME Type String**
1D Data	…aomeasurementquantity.tachovector …aomeasurement.tachovector

11.5 2D DATA

11.5.1 GENERAL DESCRIPTION

2D data are typically represented in a 2D array, where 1 dimension depends on the length of the function and the other dimension is 2 or more:

- in the most simple case the dimension is 2: one vector with Y-axis values and another one with X-axis values
- in other cases, there could be multiple X-axis (e.g. Hz and rpm)

Looking at all possible types of 2D functions that exist in the NVH Data World, and the fact that each of these functions typically needs some other parameters to describe them, the workgroup tried to simplify the handling of the data by categorizing all such functions in a few groups. These groups will be defined in ASAM ODS as different MIME types.

MIME Types for 2D Data	**MIME Type String**
Streamed	…aomeasurementquantity.streamed …aomeasurement.streamed
Referenced Streamed	…aomeasurementquantity.referencedstreamed …aomeasurement.referencedstreamed
Spectrum	…aomeasurementquantity.spectrum …aomeasurement.spectrum
Referenced Spectrum	…aomeasurementquantity.referencedspectrum …aomeasurement.referencedspectrum
Octave Spectrum	…aomeasurementquantity.octavespectrum …aomeasurement.octavespectrum
Streamed Cut	…aomeasurementquantity.streamedcut …aomeasurement.streamedcut
Referenced Streamed Cut	…aomeasurementquantity.referencedstreamedcut …aomeasurement.referencedstreamedcut
Spectrum Cut	…aomeasurementquantity.spectrumcut …aomeasurement.spectrumcut
Referenced Spectrum Cut	…aomeasurementquantity.referencedspectrumcut …aomeasurement.referencedspectrumcut
Octave Spectrum Cut	…aomeasurementquantity.octavespectrumcut …aomeasurement.octavespectrumcut

The 5 MIME types at the end of the table, with their names ending with "Cut" are necessary for 2D functions of the type "Cut". These functions are in fact the result of an operation performed on 3D objects, typically called "Waterfalls" (see next section).

These "Cut" operations result in a 2D function that can be described by the means of the 5 basic MIME types for 2D data, and a few additional parameters.

Therefore it has been decided to represent these data by a different MIME type, and to give those functions 2 Parameter Sets:

- the first Parameter Set is the one that corresponds to the Basic MIME type
- the second Parameter Set contains some few extra parameters that describe how the "Cut" operation was performed

The tables at the end of this section will describe the possible values for various parameters.

The names of the AoParameterSets are the following ones:
- for the 5 basic MIME Types: "basic"
- for the Cut MIME Type: "cut"

Any party may add company dependent parameters at any time. In order to make sure this does not lead to confusion, those parameter names must be prefixed by "_[company name]_":

e.g.:
_MBBM_SpecialWindowType
_HeadAcoustics_SpecialPsychoAcousticIndex
_LMS_SpecialModalAnalysisParameter

11.5.2 STREAMED

Parameter name	DataType enum name	Flag
creation time	DT_DATE	optional
last modification time	DT_DATE	optional
nvh_type	DT_STRING	required
acquisition/calculation method	DT_STRING	required
acoustical weighting	DT_STRING	required
amplitude scaling	DT_STRING	required
reference rpm	DT_FLOAT	optional
emphasis	DT_BOOLEAN	required
averaging mode	DT_STRING	optional
lin/exp averaging time constant	DT_FLOAT	optional
peak hold time constant	DT_FLOAT	optional
number of averages	DT_LONG	optional

The most typical example of streamed data is time domain data, sometimes also called throughput data. Mostly these are very long data streams (MegaBytes to GigaBytes).

When the averaging mode is Slow, Fast or Impulse, then the ‘lin/exp averaging time constant’ and the ‘peak hold time constant’ are to be ignored, and the values used are standardized at:

Mode	lin/exp averaging time constant	peak hold time constant
Slow	1 s	None
Fast	125 ms	None
Impulse	35 ms	1,5 s

Note that the corresponding channel name of the function is stored as instance name of AoMeasurementQuantity.

11.5.3 REFERENCED STREAMED

Parameter name	DataType enum name	Flag
creation time	DT_DATE	optional
last modification time	DT_DATE	optional
reference channel name	DT_STRING	required
reference channel quantity	reference to AoQuantity	required
nvh_type	DT_STRING	required
acquisition/calculation method	DT_STRING	required
acoustical weighting	DT_STRING	required
amplitude scaling	DT_STRING	required
reference rpm	DT_FLOAT	optional
reference rpm channel	DT_STRING (channel name)	optional
emphasis	DT_BOOLEAN	required
averaging mode	DT_STRING	optional
lin/exp averaging time constant	DT_FLOAT	optional
peak hold time constant	DT_FLOAT	optional

This is a variant of the previous data type, containing only 2 extra parameters – 'reference channel name' and 'reference channel quantity'. This is needed in the case of a cross-correlation (which is indicated through the 'nvh_type' parameter).

Name and quantity of the non-referenced channel are always documented via the corresponding base attributes of the AoMeasurementQuantity instance.

11.5.4 SPECTRUM

Parameter name	DataType enum name	Flag
creation time	DT_DATE	optional
last modification time	DT_DATE	optional
nvh_type	DT_STRING	required
acquisition/calculation method	DT_STRING	required
user acquisition/calculation method	DT_STRING	optional
window type	DT_STRING	required
user window function	reference to AoMeasurement	optional
window correction mode	DT_STRING	required
energy correction factor	DT_FLOAT	required
amplitude correction factor	DT_FLOAT	required
acoustical weighting	DT_STRING	required
amplitude scaling	DT_STRING	required
spectrum format	DT_STRING	required
averaging mode	DT_STRING	optional
exp averaging time constant	DT_FLOAT	optional
number of averages	DT_LONG	optional
reference rpm	DT_FLOAT	optional
reference rpm channel	DT_STRING (channel name)	optional

This data type is used to represent the results of operations that generate frequency or order type of data. Again, the ‘nvh_type’ field indicates the precise kind of function that is represented here.

11.5.5 REFERENCED SPECTRUM

Parameter name	DataType enum name	Flag
creation time	DT_DATE	optional
last modification time	DT_DATE	optional
reference channel name	DT_STRING	required
non-reference channel name	DT_STRING	optional
nvh_type	DT_STRING	required
acquisition/calculation method	DT_STRING	required
window type	DT_STRING	required
user window function	Reference to AoMeasurement	optional
reference window type	DT_STRING	required
reference user window function	Reference to AoMeasurement	optional
window correction mode	DT_STRING	required
energy correction factor	DT_FLOAT	required
amplitude correction factor	DT_FLOAT	required
acoustical weighting	DT_STRING	required
amplitude scaling	DT_STRING	required
spectrum format	DT_STRING	required
averaging mode	DT_STRING	optional
exp averaging time constant	DT_FLOAT	optional
number of averages	DT_LONG	optional
reference rpm	DT_FLOAT	optional
reference rpm channel	DT_STRING (channel name)	optional

This type is used in case of referenced spectra, crosspowers, etc.. It mostly needs the same attributes as the non-referenced spectrum type of data, but additionally needs information about the reference channel.

2 application relations have to be added, called "reference_channel_quantity" and "non_reference_channel_quantity", and both are references to the corresponding AoQuantity. These 2 references replace the formerly defined attributes in the table above.

11.5.6 OCTAVE SPECTRUM

Parameter name	DataType enum name	Flag
creation time	DT_DATE	optional
last modification time	DT_DATE	optional
nvh_type	DT_STRING	required
acquisition/calculation method	DT_STRING	required
acoustical weighting	DT_STRING	required
amplitude scaling	DT_STRING	required
spectrum format	DT_STRING	required
octave type	DT_LONG	required
averaging mode	DT_STRING	optional
lin/exp averaging time constant	DT_FLOAT	optional
peak hold time constant	DT_FLOAT	optional

The octave spectrum is a specific type of spectrum, since it is some kind of reduced frequency spectrum. A third-octave spectrum for instance, typically only contains 24 data values (one for each third-octave band).

11.5.7 CUT

Parameter name	DataType enum name	Flag
cut value	DT_FLOAT	required
cut width	DT_FLOAT	required

This ParameterSet is supplemental to the ParameterSet that corresponds to one of the 5 basic MIME types defined before.

This type contains the results of cuts, performed on 3D data objects (waterfalls).

The cut unit and cut width unit are annotated via the base attribute "unit" of the AoParameter instance.

11.5.8 CHOICES DEFINITIONS

The permitted values for various parameters are restricted to a set of predefined values, which are defined below. The values must be written exactly as listed below.

PARAMETER NVH_TYPE

Possible values	streamed	Ref. streamed	spectrum	Ref. spectrum	octave
Equidistant	✓	✓			
Non-equidistant	✓	✓			
Complex Spectrum			✓		
Autopower Spectrum			✓		✓
Crosspower Spectrum				✓	✓
Complex Order Spectrum			✓		
Order Autopower Spectrum			✓		
Order Crosspower Spectrum				✓	
Autocorrelation	✓				
Crosscorrelation		✓			
Frequency Response Spectrum				✓	✓
Impulse Response		✓			
Order Response Spectrum				✓	
Order Impulse Response		✓			
Shock Response Spectrum			✓		✓
Coherence				✓	✓
Order Coherence				✓	
Cepstrum	✓				

PARAMETER ACQUISITION/CALCULATION METHOD

Possible values
Undefined
Overall Level
Variable Blocked
Kalman
Sound Intensity

Possible values
Stationary Loudness
Instationary Loudness
Roughness
Fluctuation Strength
Sharpness
Articulation Index Open
Articulation Index Closed
Speech Interference Level
Impulsiveness
Principal Components Analysis
User

PARAMETER WINDOW TYPE

Possible values
Hanning
Hamming
Flattop
Uniform
Kaiser-Bessel
Blackman
Force
Exponential
Force-Exponential
User
Undefined

PARAMETER REFERENCE WINDOW TYPE

Same choices as for ‘window type’.

PARAMETER WINDOW CORRECTION MODE

Possible values
Amplitude
Energy
None

Reflects how the data are stored physically.

PARAMETER ACOUSTICAL WEIGHTING

Possible values
Linear
A
B
C
D
AB
Undefined

PARAMETER AMPLITUDE SCALING

Possible values
0-peak
Peak-peak
Magnitude
RMS

The value Magnitude is only valid for (ref) streamed data (data are rectified)

PARAMETER SPECTRUM FORMAT

Possible values
Linear
Power
PSD
ESD

These choices and the previous one (amplitude scaling) are to be interpreted in the following sense: given a sine wave with 0-peak amplitude of A, the result of a Fourier Transform is given in the next table for all relevant possible combinations of ‘amplitude scaling’ and ‘spectrum format’.

	0-peak	Peak-peak	RMS
Linear	A	2A	$\frac{A}{\sqrt{2}}$
Power	A^2	$4A^2$	$\frac{A^2}{2}$

	0-peak	Peak-peak	RMS
PSD	$\frac{A^2}{\Delta f}$	$\frac{4A^2}{\Delta f}$	$\frac{A^2}{2\Delta f}$
ESD	$\frac{A^2}{\Delta f}T$	$\frac{4A^2}{\Delta f}$	$\frac{A^2}{\Delta f}T$

Comment: For crosspowers, the spectrum format is expected to always be power. Although it is theoretically possible to take the square root of the values and the unit, this makes no real sense in terms of physics.

PARAMETER AVERAGING MODE (SPECTRUM)

Possible values
Linear
Exponential

PARAMETER AVERAGING MODE (STREAMED & OCTAVE)

Possible values
Slow
Fast
Impulse
None
Exponential
LEQ
Linear
User

The value LEQ is the abbreviation for “Equivalent Sound level”.

11.5.9 PHYSICAL STORAGE

DATA STORAGE

In principle, 3 possibilities exist to store the data:

- Data in local column(s)
- Data in external binary file – Mixed Mode
- Data in original file format external file – AoExternalReference

Especially for the first 2 cases, it is necessary to define what happens for data with multiple X-axes, because this often happens with NVH data. Although the documentation is not explicit about this, it seems that historically, nobody has ever created more than 1 independent column in a Submatrix.

Therefore, the following solution is proposed:

- All values for the X- and Y-axes are stored in one Submatrix
- An application attribute of type DT_ENUM, with name "axistype" and possible values (Xaxis, Yaxis, Both) must be added to the application element derived from AoLocalColumn.

The alternative of adding an AoParameter to the corresponding AoMeasurementQuantity has not been chosen because of performance reasons.

X-axes of type time (exactly: which has a unit that relates to the physical dimension time) are stored as 8 Byte floating point values according to IEEE 754. In case an absolute time is needed (to be associated with the first measurement value), this can be found in the "measurement_begin" base attribute of the corresponding AoMeasurement.

(REFERENCED) STREAMED DATA

Most of the time, the values are stored as floats or doubles, but sometimes, the "raw" data – as they are produced by the Analog-to-Digital converter in the measurement equipment – are stored. In that case, they are typically 16, 20 or 24 bit numbers, and in order to know what the real measured value was, a conversion needs to be performed:

*result = ((raw * scale) + offset) * calibration*

The scale, offset and calibration values are stored as generation parameters in that case.

WINDOW CORRECTION FACTORS

The window correction factors that are indicated in the corresponding parameters are supposed to be applied to the data, as defined by the window correction mode parameter.

This means that if data are annotated as amplitude corrected, and the client application wants to have the data in energy correction, then the client must perform the following operation:

*New_val = (stored_val / ampl_corr) * en_corr*

This equation is always applicable, independent of the spectrum format. So if data are stored in power format, the correction factors are the square values of the corresponding correction factors for linear format.

For referenced data, the correction factors are the multiplication/division of the referenced/non-referenced channel factors.

AOPARAMETER REFERENCES

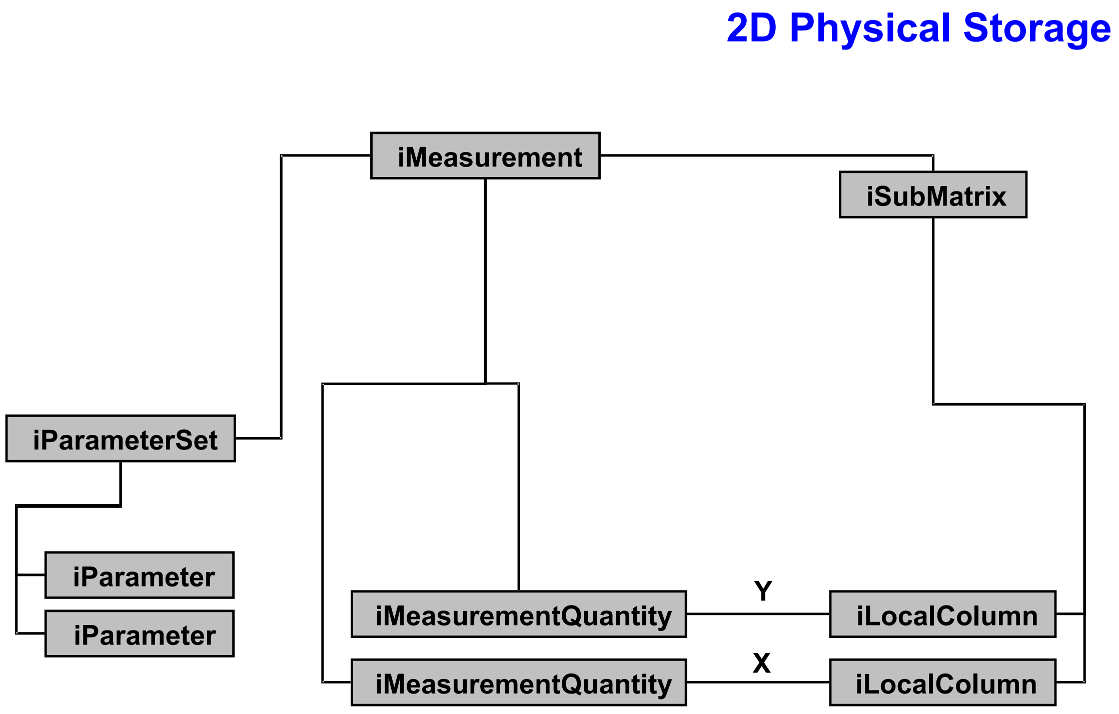
The main question for building the application model in the case of NVH data is the element from which the AoParameterSet that contains all descriptive information for the functions should be referenced. This question comes down to trading off clarity and accessibility against redundancy.

It has been decided in the workgroup that 2 possibilities will be offered: either reference from the instances of AoMeasurement or from the instances of AoMeasurementQuantity.

If two ParameterSets are found for a Measurement Quantity anyhow: one attached to the AoMeasurementQuantity instances and the other attached to the AoMeasurement instances, then the ParameterSet attached to the AoMeasurementQuantity instances has priority.

➢ *Reference from AoMeasurement instances*

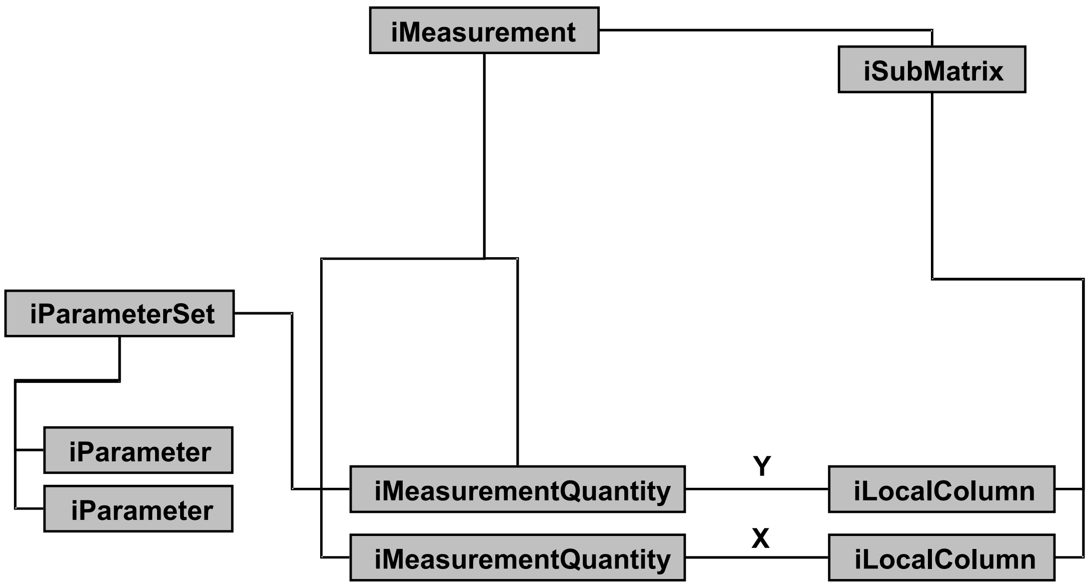
This is the best way in case all channels (AoMeasurementQuantity instances) contain the same kind of information, since redundancy would be very high in that case.



➢ *Reference from AoMeasurementQuantity instances*

This is the better way in case different kinds of functions have been used for different measurement channels (AoMeasurementQuantity instances).

2D Physical Storage

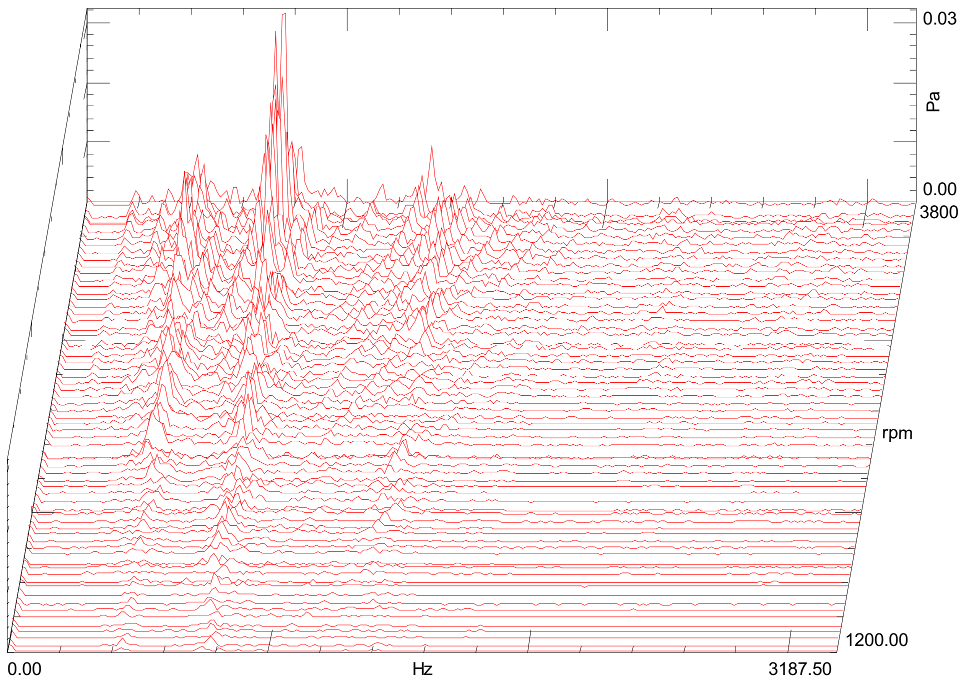


For referenced spectrum, a special case is allowed for referencing the non-reference channel. Two optional parameters are foreseen for this (non-reference channel name and non-reference quantity), but this means that this ParameterSet can only be referenced from the instances of AoMeasurementQuantity, and not from those of AoMeasurement. This can lead to quite some redundancy, and loss of performance when writing/reading the data. Therefore, it is allowed to reference the ParameterSet for such data from the AoMeasurement instances, and use application attributes instead of the 2 optional AoParameters. The application attributes should be called “non_reference_channel_name" and "non_reference_channel_quantity".

11.6 3D DATA

11.6.1 GENERAL DESCRIPTION

3D data are in fact sets of 2D data that are linked together in the 3rd dimension by an additional parameter. Typically, all X-axis of the 2D functions contained in a 3D data object are the same, but there are some cases where they are different indeed.



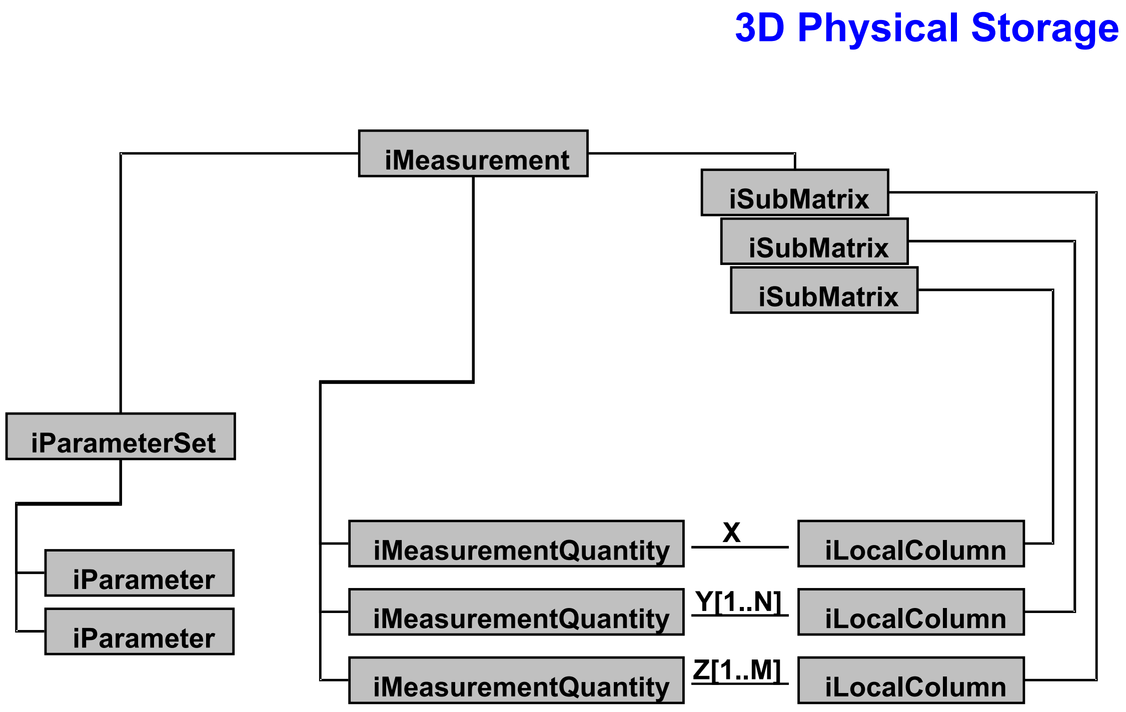
Since the 2D functions in such a 3D data object are all functions of the same kind that have been described before, the MIME types that will be introduced for 3D data are derived from the ones that were defined for 2D data.

MIME Types for 3D Data	MIME Type String
Streamed	…aomeasurementquantity.streamed3D …aomeasurement.streamed3D
Referenced Streamed	…aomeasurementquantity.referencedstreamed3D …aomeasurement.referencedstreamed3D
Spectrum	…aomeasurementquantity.spectrum3D …aomeasurement.spectrum3D
Referenced Spectrum	…aomeasurementquantity.referencedspectrum3D …aomeasurement.referencedspectrum3D
Octave Spectrum	…aomeasurementquantity.octavespectrum3D …aomeasurement.octavespectrum3D

11.6.2 PHYSICAL STORAGE

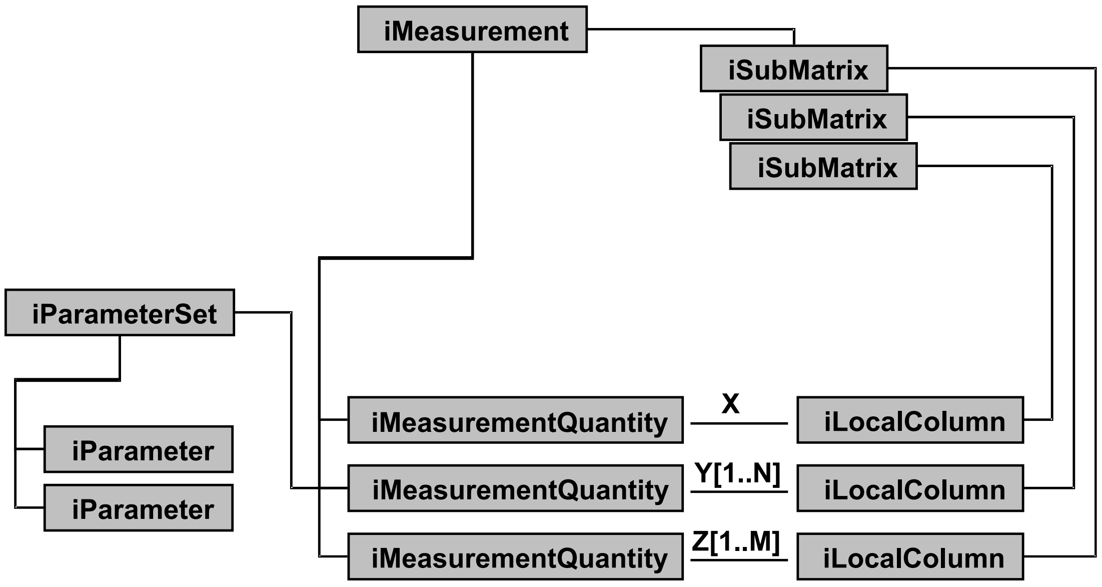
The same argument as before is true here with respect to referencing the AoParameterSet element: this can be done from the instances of AoMeasurement or from the instances of AoMeasurementQuantity, depending on the data.

- *Reference from AoMeasurement*



- *Reference from AoMeasurementQuantity*

3D Physical Storage



Following characteristics of 3D data will influence the physical storage:

- It must be possible to store an arbitrary number of 2D functions as result data for one channel.
- The 2D functions have X and Y units. These do not change with time, which means they are identical for all 2D functions within one 3D data object.
- It must be possible to assign an arbitrary number of scan values to each 2D function, these are stored as Z-axis information.
- The X value ranges of the 2D functions within one 3D data object, the X axis, need not be identical. It should be possible for the client to distinguish between 3D objects with fixed or varying X-axes.

Current ODS implementations model base elements AoSubmatrix and AoLocalColunm explicitly. This allows definition of application references between e.g. submatrices.

- Since the NVH model makes use of these possibilities, NVH data generally cannot be stored and 3D data also cannot be retrieved via the RPC-API. It is thus strongly recommended to use the OO-API for NVH data.
- This allows a uniform way for storing 3D data objects with varying and fixed X axes, which is described below.

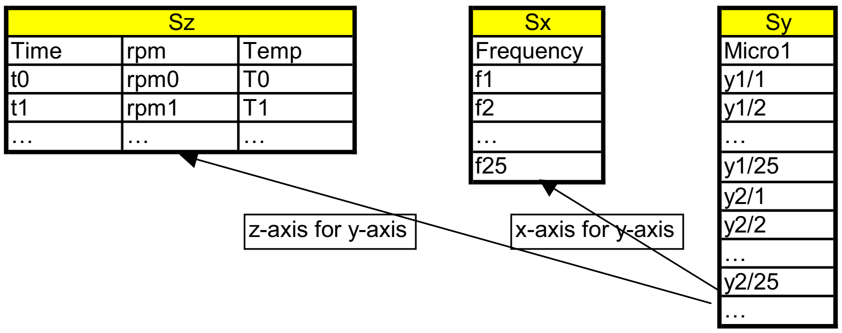
Three distinct submatrices are used to store data for one 3D data object. One submatrix (Sx) holds data for the X axes. A second submatrix (Sy) holds data for the Y axes and a third submatrix (Sz) will store the Z axes.

There could be more than one 3D data object within the context of one measurement.

Two application references between submatrix instances describe which submatrices “belong together” resp. are constituents of one 3D data object. The first is named “x-axis-for-y-axis” (inverse name “y-axis-for-x-axis”) between Sy and Sx. The second is named “z-axis-for-y-axis” (inverse name “y-axes-for-z-axes”) between Sy and Sz. It is therefore possible to navigate to the X or Z-axes, if a fast channel (Y-axes) is given.

The following two figures show how 3D data objects with fixed and varying X axes are stored in local columns and submatrices.

Fixed X-axis:

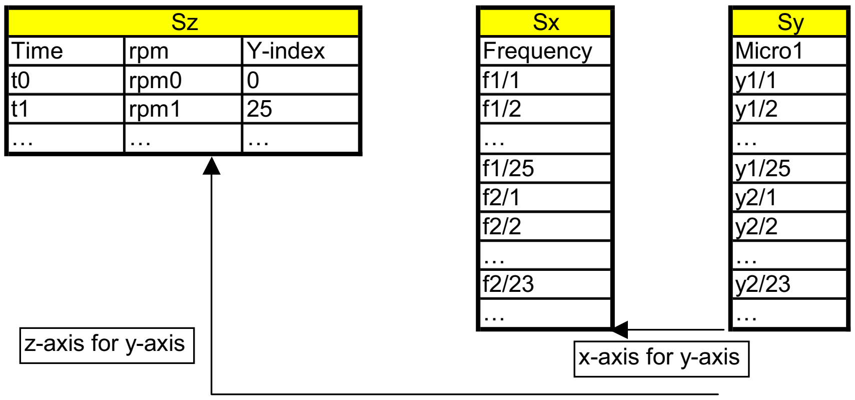


Sz		
Time	rpm	Temp
t0	rpm0	T0
t1	rpm1	T1
…	…	…

Sx
Frequency
f1
f2
…
f25

Sy
Micro1
y1/1
y1/2
…
y1/25
y2/1
y2/2
…
y2/25
…

Varying X-axis:



Sz		
Time	rpm	Y-index
t0	rpm0	0
t1	rpm1	25
…	…	…

Sx
Frequency
f1/1
f1/2
…
f1/25
f2/1
f2/2
…
f2/23
…

Sy
Micro1
y1/1
y1/2
…
y1/25
y2/1
y2/2
…
y2/23
…

It is possible that e.g. rpm has also been measured as independent streamed data, and thus has an AoParameterSet connected to the AoMeasurementQuantity for the local column that represents the Y values of the rpm signal. If, in the same measurement, 3D data objects have been stored, with rpm as tracking value, then the Z-axis of these 3D objects is connected to the same AoMeasurementQuantity, and as such, even the Z-axis will have an AoParameterSet connected to it. This is however not the case in general.

It is not an obligation to have an AoParameterSet for Z-axis local columns.

It must be possible to extract the distinct 2D functions from the column, which contains data for the fast channel.
For fixed X-axes, the number of Y values of a 2D function is given by the number of X-axes values nx (number of rows Sx). The nth Y-axis can therefore be read by querying values

[(n – 1) * nx, n * nx – 1] from Sy

For varying X-axes, a special local column named “yindex” must be created in Sz. The client can read the complete column in one request and find the row range depending on the Z value of the corresponding 2D function in Sy and Sz: [yindex[n-1], yindex[n]-1]; for the last 2D function: [yindex[n-1],ny]

The following constraints must be fulfilled:

- for fixed X-axis:

  ny = nz * nx

- for varying X-axis:

  ny = nx
  yindex[n] > yindex[m] if n > m
  yindex[n] < ny for all n

11.7 THE MTL DATA TYPE

11.7.1 GENERAL DESCRIPTION

MTL stands for “**M**ultidimensional **T**ime at **L**evel”, a statistical method for the time series measurement domain. It is a kind of multi-dimensional classification, representing the total amount of time the measured signals are on the specified levels.

An MTL calculation is performed on a group of channels of a time series measurement (typically up to 6 channels, each up to 1 Million values) and results in one submatrix containing the classified quantities: one independent channel for the levels and for each quantity, there is a dependent channel containing the time information. For a full multidimensional time at level information there are two information parts reserved. They have to be used:

MIME Types for MTL Data	MIME Type String
MTL Description	…aomeasurement.mtl
MTL Result	…aosubmatrix.mtl

OVERALL AOPARAMETER

Because of the product dependencies in calculation and handling the vendor and product information should be part of the overall data information.

Parameter name	DataType enum name	Flag	Comment
Vendor	DT_STRI NG	optional	Vendor of the software which created the AoParameterSet
Product	DT_STRI NG	optional	Product name which created the AoParameterSet
ProductVersion	DT_STRI NG	optional	Version of the product
CreationDate	DT_DATE	optional	Creation date of the AoParameterSet

RESULT SPECIFIC AOPARAMETER

Following attributes are required/interesting for correct interpretation of the result data.

Parameter name	DataType enum name	Flag	Comment
dimension	DT_LONG	required	Dimension of the classification corresponding to the number of classification channels (2..n)
ClassMinimum	DT_STRING	required	Consists of all Classification Minimum (bottom of classification ranges) of all classification channels, according to ODS definitions (CSV), e.g. “-2.34,-3.45,4.34”
ClassMaximum	DT_STRING	required	Consists of all Classification Maximum (top of classification ranges) of all classification channels, according to ODS definitions (CSV), e.g. “2.34,3.45,14.34”
NClasses	DT_STRING	required	Consists of all number of classes of all classification channels, according to ODS definitions (CSV), e.g. “10,10,64”. BinWidth(i) = (ClassMaximum(i)-ClassMinimum(i))/NClasses(i)
BinWidth	DT_STRING	optional	Consists of all BinWidths of all classification channels, according to ODS definitions (CSV) BinWidth(i) = (ClassMaximum(i)-ClassMinimum(i))/NClasses(i)
SourceReference	DT_STRING	optional	URL / ASAMPath of the source
SourceLength	DT_STRING	optional	Length of the source as String
SourceMinimum	DT_STRING	optional	Consists of all Minimum of all source channels of the time series, according to ODS definitions (CSV), e.g. “-2.34,-3.45,4.34”. The matching AoUnit has to be referenced by this AoParameter
SourceMaximum	DT_STRING	optional	Consists of all Maximum of all source channels of the time series, according to ODS definitions (CSV), e.g. “-2.34,-3.45,4.34”. The matching AoUnit has to be referenced by this AoParameter
Description	DT_STRING	optional	A short description

11.7.2 PHYSICAL STORAGE

CONTENT AND REFERENCES

The MTL Description is the container for one or more multi-dimensional time at level result data (it depends on the writing application if there is an AoMeasurement per result or not).

It is referred by:

- one AoParameterSet with the MimeType “mtl” (where all AoParameters refer to)
- every “overall” AoParameter directly

It refers to:

- the source AoMeasurement time series if possible.

Every AoParameter of all multidimensional time at level calculations has to refer to the corresponding AoParameterSet (1:n) and additionally / optionally to the corresponding following ODS-entities:

- AoMeasurement (parameter for all time at level calculations inside this AoMeasurement)
- AoSubmatrix (parameter for one time at level calculation result only)
- AoMeasurementQuantity (if it is a parameter of that type)
- AoUnit (for the corresponding unit of the parameter value)

Each MTL Result itself is handled by one AoSubmatrix.

This part contains the so called multidimensional time at level result. It includes exactly one multidimensional time at level result of a group of time series channels (up to 6 channels) and handles the full information necessary for a correct interpretation of the data.

It has to have references to:

- the AoMeasurement it belongs to

It is referred by

- 2 AoLocalColumns per dimension (== classified channel)
- the AoLocalColumns for the summarized results (time, rotations)
- the AoParameter used for this mtl (see result specific AoParameter above)

These references have to follow some rules:

- 2 AoLocalColumns per dimension (==classified channel)
  There are 2 AoLocalColumn (2 AoMeasurementQuantities) for each classified channel. They shall have the link to and the name of the source channel’s AoMeasurementQuantity they are calculated from, extended by “.Bin” or “.CenterValue”:
  - `<ChannelName>.Bin`: This AoLocalColumn contains the bin number of the classified channel. It is an AoMeasurementQuantity with the AoUnit “”.
  - `<ChannelName>.CenterValue`: This AoLocalColumn contains the level information the bin stands for. Unlike other classifications like rainflow it is the CenterValue of the bin. It is an AoMeasurementQuantity with the source channel’s AoUnit.

  The classification attributes of these AoLocalColums like “ClassMinimum”, “ClassMaximum” or “NClasses” will be handled by AoParameter of the AoSubmatrix (see section "Result Specific AoParameter" above).
- the AoLocalColumns for the summarized results (time, rotations, ...)

These AoLocalColums shall have the link to the correct AoQuantity (e.g. “rotation” or “time”) and the corresponding AoUnit (e.g. “rad” or “sec”).
Only the time channel is required, the others (like rotation) are optional.

- any AoParameter used for this mtl and its parts
  The Parameters of the calculation will be handled by AoParameters and AoParameterSets. For further information please look at section "Result Specific AoParameter" above.

EXAMPLE:

EXAMPLE FOR AN AOSUBMATRIX:

The channel SumRotation is optional.

```
F_x.Bin F_x.CenterValue F_y.Bin F_y.CenterValue SumTime   SumRotation
1       1               1       -9              2.098      1.987
1       1               2       -7              3.98       3.87
2       3               1       -9              2.12       2.76
...
...
...
```

11.8 THE RAINFLOW DATA TYPE

11.8.1 GENERAL DESCRIPTION

A Rainflow classification is a statistical method to reduce a time domain signal to a minimal size, without losing the durability information contained in the signal.

A Rainflow calculation is normally performed on one channel of a time series measurement (one AoLocalColumn). In specialized products it is also possible to calculate new rainflow classifications out of others. This type of calculation results in one 3D matrix (integer Z over integer X over integer Y) and optionally 1 vector of values, called the residuum (integer X values). The typical size of the result matrix is between 3000 and 20000 values. For a full rainflow information there are three information parts reserved (Description and Matrix have to be used, Residuum is optional):

MIME Types for Rainflow Data	**MIME Type String**
Rainflow Description	…aomeasurement.rainflow
Rainflow Result Matrix	…aosubmatrix.rainflow.matrix
Rainflow Result Residuum	…aosubmatrix.rainflow.residuum

OVERALL AOPARAMETER

Because of the product dependencies in calculation and handling the vendor and product information should be part of the overall data information.

Parameter name	**DataType enum name**	**Flag**	**Comment**
Vendor	DT_STRING	optional	Vendor of the software which created the AoParameterSet
Product	DT_STRING	optional	Product name which created the AoParameterSet
ProductVersion	DT_STRING	optional	Version of the product
CreationDate	DT_DATE	optional	Creation date of the AoParameterSet

RESULT SPECIFIC AOPARAMETER FOR RAINFLOW.MATRIX

Following attributes are required/interesting for correct interpretation of the result data.

Parameter name	DataType enum name	Flag	Comment
ClassMinimum	DT_FLOAT	required	Minimum of the classification range. The matching AoUnit has to be referenced by this AoParameter
ClassMaximum	DT_FLOAT	required	Maximum of the classification range. The matching AoUnit has to be referenced by this AoParameter
NClasses	DT_SHORT	required	Number of classes between ClassMinimum and ClassMaximum. Range 1..128
AmplitudeSuppression	DT_FLOAT	required	Range of the filter in ClassWidths Range 1..NClasses
ResiduumSubmatrix	DT_STRING	required	String of Name of residuum submatrix + ";" + "Version": The reference to the optional residuum submatrix. "" if no residuum is available
Symmetric	DT_BOOLEAN	optional	Default and therefore If not available or not set it is TRUE
SourceReference	DT_STRING	optional	URL / ASAMPath of the source
SourceMinimum	DT_FLOAT	optional	Minimum of the source data of the time series. The matching AoUnit has to be referenced by this AoParameter
SourceMaximum	DT_FLOAT	optional	Maximum of the source data of the time series. The matching AoUnit has to be referenced by this AoParameter
SourceTimeLength	DT_FLOAT	optional	Time length of the time series. The matching AoUnit has to be referenced by this AoParameter
MaxClass	DT_LONG	optional	Maximum class with counts > 0 Range 1..NClasses
MinClass	DT_LONG	optional	Minimum class with counts > 0 Range 1..NClasses
TotalCount	DT_LONG	optional	Sum of all Counts Range 0..2^{64}
NormalizingFactor	DT_FLOAT	optional	The factor for normalizing to a standard length (not used by all products)
ReferenceLength	DT_FLOAT	optional	The length of the classification compared to others (not used by all products). If used the matching AoUnit (e.g. "km") has to be referenced by this AoParameter

RESULT SPECIFIC AOPARAMETER FOR RAINFLOW.RESIDUUM

Parameter name	DataType enum name	Flag	Comment
rainflowsubmatrix	DT_STRING	required	String of Name of rainflow submatrix + “;” + “Version”: The reference to the rainflow submatrix.

11.8.2 PHYSICAL STORAGE

CONTENT AND REFERENCES

The Rainflow Description is the container for one or more rainflow result data (it depends on the writing application if there is an AoMeasurement per result or not).

It is referred by:
- one AoParameterSet with the MimeType “rainflow” (where all AoParameters refer to)
- every “overall” AoParameter directly

It refers to:
- the source AoMeasurement time series if possible.

Every AoParameter instance of all rainflow calculations has to refer to the corresponding AoParameterSet instance (1:n) and additionally / optionally to instances of the corresponding following ODS-entities:
- AoMeasurement (if it is a parameter for all rainflow-submatrices)
- AoSubmatrix (if it is a parameter for one rainflow calculation result only)
- AoMeasurementQuantity (if it is a parameter of that type)
- AoUnit (for the corresponding unit of the parameter value)

Each Rainflow Result itself is handled by one (or optionally two) instances of AoSubmatrix.

The **first part** contains the so called n*n (n<=128) rainflow matrix result data in an AoSubmatrix. It includes exactly one rainflow matrix and handles the full information necessary for a correct interpretation of the data (for the optional “residuum” see below).

It has to have references to the
- AoMeasurement it belongs to
- AoMeasurementQuantity of the source (the original AoMeasurementQuantity the calculation is done from) if possible

It is referred by
- AoLocalColumns
- any AoParameter used for this rainflow classification matrix

Inside this AoSubmatrix of type “rainflow.matrix ”there are 3 instances of AoLocalColumn (containing data for 3 instances of AoMeasurementQuantity):

- "From": This AoLocalColumn contains the x-axis values of the matrix as independent channel. It is an AoMeasurementQuantity without an AoUnit [1].
- "To": This AoLocalColumn contains the y-axis values of the matrix as independent channel. It is an AoMeasurementQuantity without an AoUnit [1].
- "Count": This AoLocalColumn contains the z-axis values of the matrix. It is an AoMeasurementQuantity without an AoUnit [1].

The values inside the matrix are not sorted in any way.

EXAMPLE:

interpretation: "In the origin time series signal there were 212 oscillations from class 1 to class 6, 110 oscillations from class 4 to class 17 and .... "

```
FROM     TO        COUNT
1        6         212
4        17        110
5        34        65
..       ..        ..
```

The **second part** of the rainflow result is optional. Only a few products are able to handle it, others don't. It contains the so called rainflow residuum in an AoSubmatrix of exactly one rainflow classification.

It has to have references to the
- AoMeasurement it belongs to
- AoMeasurementQuantity of the source (the origin AoMeasurement-Quantity the classification is calculated from) if possible

It is referred by
- 1 AoLocalColumn
- any AoParameter used for this rainflow classification residuum

Inside the AoSubmatrix of type "rainflow.residuum" there is one instance of AoLocalColumn (containing values for 1 instance of AoMeasurementQuantity):

- "Residuum": This AoLocalColumn contains the values of the residuum. It is an AoMeasurementQuantity without an AoUnit [1].
  The maximum length is (NClasses*2)-1.

The values inside the matrix are not sorted in any way.

[1] attention with handling units:
The AoLocalColumn "From" and the "To" contain the X- and Y-coordinates of the counts inside the rainflow matrix. The AoLocalColumn "Residuum" contains the class of the uncountable point inside the residuum.
Because it is a widely used way to save and handle the values in a non-existing unit "classes" the real values have to be calculated / calibrated via implicit knowledge. A generic client doesn't know about that and only gets the value 1...128 without any unit. A special client knowing about the rainflow MimeType has to support the definitions given to interpret the values in the right way.

EXAMPLE:

interpretation: "In the origin timeseries there were some not countable oscillations: class 1, class 6, class 4, 17 and .... "

```
RESIDUUM
     1
     6
     4
     17
         ..
```

Finding the corresponding submatrices:

One rainflow submatrix and the corresponding residuum matrix can easily be found via the standardized AoParameter.

Every AoSubmatrix of type "..rainflow.residuum" has to have a corresponding AoSubmatrix of type "..rainflow.matrix".

CALCULATION/CALIBRATION OF VALUES: FROM "CLASSES" TO UNITS

The values in the AoMeasurementQuantity "From", "To" and "Count" are not directly useable because of the handling in a non-existing unit "classes".

So any generic client cannot handle the implicit knowledge of the classification data and will only show the values 1..NClasses (of classes) without any explicit unit.

To get the correct values out of the data the calculation / calibration has to do the following steps:

- First way: Only possible with source AoMeasurementQuantity
  - get the AoSubmatrix "..rainflow.matrix" of the AoMeasurement
  - get the referred AoMeasurementQuantity
  - get the Unit referred by it

- Second way: Always possible
  - get the AoSubmatrix "..rainflow.matrix" of the AoMeasurement
  - from this AoSubmatrix get all AoParameter referencing it
  - from these AoParameter get the value of the AoParameter "ClassMinimum" or "ClassMaximum"
  - from one of these AoParameter get the reference to the AoUnit
  - get the AoUnit

- Calculate the ClassWidth, ClassBorders, etc. and the values in Units as needed out of the AoParameter "ClassMinimum", "ClassMaximum", "NClasses", ...

11.9 THE TAL DATA TYPE

11.9.1 GENERAL DESCRIPTION

TAL stands for "**T**ime **A**t **L**evel", a widely used statistical method for the time series measurement domain. It is a kind of classification, representing the total time a signal resides on a specific level.

A TAL calculation is typically performed on one channel of a time series measurement (one AoLocalColumn), and results in one vector containing the classified result as floats plus an additional independent channel for the level. (also float values) The size of the result is typically up to 1000 values per channel. For a full time at level information there are two information parts reserved. They have to be used:

MIME Types for MTL Data	MIME Type String
TAL Description	…aomeasurement.tal
TAL Result	…aosubmatrix.tal

OVERALL AOPARAMETER

Because of the product dependencies in calculation and handling the vendor and product information should be part of the overall data information.

Parameter name	DataType enum name	Flag	Comment
Vendor	DT_STRING	optional	Vendor of the software which created the AoParameterSet
Product	DT_STRING	optional	Product name which created the AoParameterSet
ProductVersion	DT_STRING	optional	Version of the product
CreationDate	DT_DATE	optional	Creation date of the AoParameterSet

RESULT SPECIFIC AOPARAMETER

Following attributes are required/interesting for correct interpretation of the result data.

Parameter name	DataType enum name	Flag	Comment
ClassMinimum	DT_FLOAT	required	Minimum of the classification range. The matching AoUnit has to be referenced by

Parameter name	DataType enum name	Flag	Comment
			this AoParameter
ClassMaximum	DT_FLOAT	required	Maximum of the classification range. The matching AoUnit has to be referenced by this AoParameter
NClasses	DT_SHORT	required	Number of classes between ClassMinimum and ClassMaximum. Range 1..100
SourceReference	DT_STRING	optional	URL / ASAMPath of the source
SourceMinimum	DT_FLOAT	optional	Maximum of the source data of the time series. The matching AoUnit has to be referenced by this AoParameter
SourceMaximum	DT_FLOAT	optional	Minimum of the source data of the time series. The matching AoUnit has to be referenced by this AoParameter
MaxClass	DT_LONG	optional	Maximum class with counts > 0. Range 0..NClasses
MinClass	DT_LONG	optional	minimum class with counts > 0. Range 0..NClasses
TotalCount	DT_LONG	optional	Sum of all Counts
NormalizingFactor	DT_FLOAT	optional	The factor for normalizing to a standard length (not used by all products)
ReferenceLength	DT_FLOAT	optional	The length of the classification compared to others (not used by all products). If used the matching AoUnit (e.g. “km”) has to be referenced by this AoParameter

11.9.2 PHYSICAL STORAGE

CONTENT AND REFERENCES

The TAL Description is the container for one or more time at level result data (it depends on the writing application if there is an AoMeasurement per result or not).

It is referred by:

- one AoParameterSet with the MimeType "tal" (where all AoParameters refer to)
- every "overall" AoParameter directly

It refers to:

- the source AoMeasurement time series if possible.

Every AoParameter of all time at level calculations has to refer to the corresponding instance of AoParameterSet (1:n) and additionally / optionally to the corresponding instances of following ODS-entities:

- AoMeasurement (parameter for all time at level calculations inside this AoMeasurement)
- AoSubmatrix (parameter for one time at level calculation result only)
- AoMeasurementQuantity (if it is a parameter of that type)
- AoUnit (for the corresponding unit of the parameter value)

Each TAL Result itself is handled by one AoSubmatrix.

This part contains the so called time at level result data in an AoSubmatrix. It includes exactly one time at level of one time series channel and handles the full information necessary for a correct interpretation of the data.

It has to have references to the

- AoMeasurement it belongs to
- AoMeasurementQuantity of the source (the original AoMeasurementQuantity the calculation is done from) if possible

It is referred by

- 2 AoLocalColumns
- any AoParameter used for this tal

Inside this AoSubmatrix of type ".....tal" there have to be 2 instances of AoLocalColumn (containing the values of 2 instances of AoMeasurementQuantity). One AoLocalColumn instance is the independent channel "level" and is optionally of type explicit or of type implicit:

- "TAL": This AoLocalColumn contains the time at level information. It is an AoMeasurementQuantity with the AoUnit "sec". The MimeType is "....LocalColumn.tal"
- "level": This AoLocalColumn contains the level information and is the independent channel of the AoSubmatrix. It is an AoMeasurementQuantity with the same AoUnit as the source channel the result is calculated from.
  This AoLocalColumn can be either of type implicit (where the beginning and delta is defined and the values are calculated online) or of type explicit (the values are explicit in the submatrix). The MimeType is "....LocalColumn.tal.level"

The channel names inside analyzing applications should be used by concatenating e.g. the name of the submatrix +”.” + the channel name to avoid confusion when using more than one “tal” result.

EXAMPLE: **“FORCEX.TAL”**

```
TAL      level (implicit/explicit)
0.001    0.00
0.004    0.1
0.01     0.2
..       ..
```

11.10 QUANTITIES AND UNITS

This section specifies:

- The minimum set of MIME types for Quantities: for each of them, at least one quantity of this MIME type should be available in every application model for NVH Data
- The minimum set of units: the default units for the quantities that are proposed in the previous part

11.10.1 QUANTITIES

GENERAL NVH QUANTITIES

…aoquantity.rotationalspeed
…aoquantity.order
…aoquantity.vibrationacceleration
…aoquantity.vibrationvelocity
…aoquantity.vibrationdisplacement
…aoquantity.angularacceleration
…aoquantity.angularvelocity
…aoquantity.angulardisplacement
…aoquantity.angle
…aoquantity.crankangle
…aoquantity.percent
…aoquantity.phase
…aoquantity.mechanicalimpedance
…aoquantity.admittance
…aoquantity.stiffness
…aoquantity.flexibility
…aoquantity.accelerance
…aoquantity.charge
…aoquantity.octavefrequency
…aoquantity.microstrain

SOUND QUALITY QUANTITIES

…aoquantity.soundpressure
…aoquantity.soundpower
…aoquantity.soundintensity
…aoquantity.reveberationtime
…aoquantity.loudness
…aoquantity.specificloudness
…aoquantity.sharpness
…aoquantity.roughness
…aoquantity.specificroughness
…aoquantity.fluctuationstrength
…aoquantity.specificfluctuationstrength
…aoquantity.tonality
…aoquantity.annoyance
…aoquantity.pleasentness
…aoquantity.impulsiveness

…aoquantity.specificimpulsiveness
…aoquantity.articulationindex
…aoquantity.degreeofmodulation
…aoquantity.compressedpressure
…aoquantity.quefrency
…aoquantity.speechintellegibility
…aoquantity.fluctuationstrength

11.10.2 UNITS

The following table contains the minimum set of units for the NVH-related quantities.

Quantity	Unit	factor	offset	Physical Dimension [len, mas, tim, cur, temp, mol, lum]
…aoquantity.rotationalspeed	rpm	1/60	0	[0,0,-1,0,0,0,0]
…aoquantity.order	ord	1	0	[0,0,0,0,0,0,0]
acceleration, aoquantity.vibrationacceleration	m/s^2	1	0	[1,0,-2,0,0,0,0]
Velocity, aoquantity.vibrationvelocity	m/s	1	0	[1,0,-1,0,0,0,0]
Length, aoquantity.vibrationdisplacement	m	1	0	[1,0,0,0,0,0,0]
aoquantity.angularacceleration	deg/s^2	pi/180	0	[0,0,-2,0,0,0,0]
aoquantity.angularacceleration	rad/s^2	1	0	[0,0,-2,0,0,0,0]
aoquantity.angularvelocity	rad/s	1	0	[0,0,-1,0,0,0,0]
aoquantity.angularvelocity	deg/s	pi/180	0	[0,0,-1,0,0,0,0]
Radian, aoquantity.angulardisplacement	rad	1	0	[0,0,0,0,0,0,0]
Degree, aoquantity.angulardisplacement, aoquantity.angle, aoquantity.crankangle	deg	pi/180	0	[0,0,0,0,0,0,0]
Percent	%	1	0	[0,0,0,0,0,0,0]
Phase	deg	pi/180	0	[0,0,0,0,0,0,0]
aoquantity.mechanicalimpedance	N/(m/s)	1	0	[0,1,-1,0,0,0,0]
aoquantity.admittance	S	1	0	[-2,-1,3,2,0,0,0]
aoquantity.stiffness	N/m	1	0	[0,1,-2,0,0,0,0]
aoquantity.flexibility	M/N	1	0	[0,-1,2,0,0,0,0]
aoquantity.accelerance	(m/s^2)/N	1	0	[0,-1,0,0,0,0,0]
aoquantity.charge	PC	1e-12	0	[0,0,1,1,0,0,0]
aoquantity.octavefrequency	Hz	1	0	[0,0,-1,0,0,0,0]
Pressure, aoquantity.soundpressure	Pa	1	0	[-1,1,-2,0,0,0,0]
Power, aoquantity.soundpower	W	1	0	[2,1,-3,0,0,0,0]
Intensity, aoquantity.soundintensity	W/m^2	1	0	[0,1,-3,0,0,0,0]
Time, aoquantity.reverberationtime	S	1	0	[0,0,1,0,0,0,0]

Quantity	Unit	factor	offset	Physical Dimension [len, mas, tim, cur, temp, mol, lum]
aoQuantity.loudness	sone	1	0	[0,0,0,0,0,0,0]
aoquantity.specificloudness	sone/bark	1	0	[0,0,1,0,0,0,0]
aoquantity.sharpness	acum	1	0	[0,0,0,0,0,0,0]
aoquantity.roughness	asper	1	0	[0,0,0,0,0,0,0]
aoquantity.specificroughness	asper/bark	1	0	[0,0,1,0,0,0,0]
aoquantity.fluctuationstrength	vacil	1	0	[0,0,0,0,0,0,0]
aoquantity.specificfluctuationstrength	vacil/bark	1	0	[0,0,1,0,0,0,0]
aoquantity.tonality	Tu	1	0	[0,0,0,0,0,0,0]
aoquantity.annoyance	au	1	0	[0,0,0,0,0,0,0]
aoquantity.pleasentness	pu	1	0	[0,0,0,0,0,0,0]
aoquantity.impulsiveness	iu	1	0	[0,0,0,0,0,0,0]
aoquantity.apecificimpulsiveness	iu/bark	1	0	[0,0,1,0,0,0,0]
aoquantity.articulationindex	%	100	0	[0,0,0,0,0,0,0]
aoquantity.degreeofmodulation	%	100	0	[0,0,0,0,0,0,0]
aoquantity.compressedpressure	CPa	1	0	[0,0,0,0,0,0,0]
aoquantity.quefrency	S	1	0	[0,0,1,0,0,0,0]
aoquantity.speechintellegibility	%	100	0	[0,0,0,0,0,0,0]
Temperature	T	1	0	[0,0,0,0,1,0,0]
Torque	N*m	1	0	[2,1,-2,0,0,0,0]
Density	kg/m^3	1	0	[-3,1,0,0,0,0,0]
aoquantity.frequencygroup	bark	1	0	[0,0,-1,0,0,0,0]
Frequency	Hz	1	0	[0,0,-1,0,0,0,0]
Force	N	1	0	[1,1,-2,0,0,0,0]

The following table contains the combination of units, needed to store transfer, cps, aps data:

Quantity	Unit	factor	offset	Physical Dimension [len, mas, tim, cur, temp, mol, lum]
Force	N^2	1	0	[2,2,-4,0,0,0,0]
Pressure, aoquantity.soundpressure	Pa^2	1	0	[-2,2,-4,0,0,0,0]
Acceleration, aoquantity.vibrationacceleration	m^2/s^4	1	0	[2,0,-4,0,0,0,0]
Velocity, aoquantity.vibrationvelocity	m^2/s^2	1	0	[2,0,-2,0,0,0,0]
Intensität, aoquantity.soundintensity	W^2/m^4	1	0	[0,2,-6,0,0,0,0]

Quantity	Unit	factor	offset	Physical Dimension [len, mas, tim, cur, temp, mol, lum]
Power, aoquantity.soundpower	W^2	1	0	[4,2,-6,0,0,0,0]
Acceleration(aoquantity.vibrationacceleration) / Pressure(aoquantity.soundpressure)	m/Pa*s^2	1	0	[2,-1,0,0,0,0,0]
Pressure(aoquantity.soundpressure) / Acceleration(aoquantity.vibrationacceleration)	Pa*s^2/m	1	0	[-2,1,0,0,0,0,0]
Acceleration(aoquantity.vibrationacceleration) * Velocity(aoquantity.vibrationvelocity)	m^2/s^3	1	0	[2,0,-3,0,0,0,0]
Acceleration(aoquantity.vibrationacceleration)* Pressure(aoquantity.soundpressure)	Pa*m/s^2	1	0	[0,1,-4,0,0,0,0]
Acceleration(aoquantity.vibrationacceleration)* Force	N*m/s^2	1	0	[2,1,-4,0,0,0,0]
Velocity(aoquantity.vibrationvelocity) * Pressure(aoquantity.soundpressure)	Pa*m/s	1	0	[0,1,-3,0,0,0,0]
Velocity(aoquantity.vibrationvelocity) * Force	N*m/s	1	0	[2,1,-3,0,0,0,0]
Pressure(aoquantity.soundpressure) * Force	Pa*N	1	0	[0,2,-4,0,0,0,0]
Pressure(aoquantity.soundpressure) / Velocity(aoquantity.vibrationvelocity)	Pa*s/m	1	0	[-2,1,-1,0,0,0,0]
Force / Velocity(aoquantity.vibrationvelocity)	N*s/m	1	0	[0,1,-1,0,0,0,0]
Velocity(aoquantity.vibrationvelocity) / Pressure(aoquantity.soundpressure)	m/Pa*s	1	0	[2,-1,1,0,0,0,0]
Velocity(aoquantity.vibrationvelocity) / Force	m/N*s	1	0	[0,-1,1,0,0,0,0]

11.11 REVISION HISTORY

Date Editor	Changes
2003-03-29 G. Sablon	Created.
2003-04-10 G. Sablon, S. Wartini, K. Rucker, E. Klinkenberg	Modification based on workgroup meeting
2003-04-24 G. Sablon	Completion to send out for approval
2003-04-30 G. Sablon	Implement changes after proofreading by NVH workgroup
2003-05-09 G. Sablon	Implement changes after ASAM-ODS workgroup meeting
2003-10-14 R. Bartz	Some errors have been fixed Sections on TAL, MTL, PSD have been merged into the chapter; several errors in them have been fixed
2003-10-17	The FTR meeting agreed to the current text with two modifications required
2003-10-02 G. Sablon	Substituted the PSD part by the Rainflow part.
2003-12-24 G. Sablon	Changed the structure of the MTL, Rainflow and TAL parts to something more in line with the rest of this Chapter.
2003-12-24 G. Sablon	Added 2 attributes for referenced spectrum (reference to non-reference data) and some comments on physical storage for this.
2003-12-30 R. Bartz	Some minor textual and formatting changes have been introduced The **Release** version has been created
2004-02-14 R. Bartz	Included section 11.9 (List of NVH quantities and units) as specified by the NVH workgroup and delivered by K. Rucker
2004-05-12 K.Rucker, F. Neyrinck	Changed data type of various parameters (DT_ENUM → DT_STRING) Figures reworked

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 12
CALIBRATION DATA MODEL

Version 1.0

Association for **S**tandardization of
Automation and **M**easuring Systems

Dated: 30.09.2004

Status of Document

Reference:	ASAM ODS Version 5.0 Calibration Data Model
Date:	30.09.2004
Author:	Dr. Bruno Thelen, Schenck Pegasus
Type:	Specification
Doc-ID:	ASAM_ODS_50_CH12_Calibration_Model.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 (odsftr@asam.net) to make sure this issue will be addressed within the next review cycle.

Contents

12 THE ASAM ODS CALIBRATION DATA MODEL 12-3

12.1 INTRODUCTION 12-3
12.2 CALIBRATION DATA IN THE TEST FIELD CONTEXT 12-4
12.2.1 THE IMPACT OF THE MEASURING EQUIPMENT ADMINISTRATION 12-4
12.2.2 THE IMPACT OF THE CALIBRATION LOCATION 12-6
12.2.3 THE IMPACT OF THE CURRENT ODS-BASE MODEL 12-7
12.3 THE CALIBRATION PROCESS IN GENERAL 12-8
12.4 THE CALIBRATION APPLICATION MODEL 12-9
12.4.1 ADMINISTRATIVE APPLICATION ELEMENTS 12-9
12.4.2 MEASURING DATA APPLICATION ELEMENTS 12-12
12.4.3 DESCRIPTIVE DATA APPLICATION ELEMENTS 12-14
12.4.4 THE COMPLETE CALIBRATION DATA MODEL 12-18
12.5 REVISION HISTORY 12-21

Scope

This document describes the application model for calibration data of ASAM ODS Version 5.0.

Intended Audience

This document is intended for implementers of ASAM ODS Version 5.0. It shall be used as a technical reference on how to store calibration information in 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

12 THE ASAM ODS CALIBRATION DATA MODEL

12.1 INTRODUCTION

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 calibration data model is based on the ASAM ODS base model, and is intended to be a minimal application model. This means that anyone who creates a company specific application model can use the proposed model and add company specific items.

Until now the main usage of an ASAM ODS server is restricted to storing and retrieving 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.

As the operating quality of measuring components shifts by the time (influenced by temperature, humidity, ... etc.), the quality of the sampled measuring data is affected. A honest interpretation of measuring data should include the introspection of the test bed operating conditions at the time the data were collected. Generally speaking, that is a) the testing context, b) the test bed components configuration and c) the associated calibration status. Hence a measuring data archive is not complete before the testing context and its status is documented.

To sum up there are two main aspects where the archived test bed configuration, its calibration status and calibration data are of additional value:

- a data analysis is done at a time where the complete test bed arrangement has changed due to other experiments on the same test bed.
- automated, and contiguous supervision of the test bed operating status guarantees an equivalent lasting quality level of the complete test bed equipment.

The following picture presents the typical activity cycle for a continued quality verification and certification process. The process cycle starts with the documentation of the test bed equipment and its calibration status and ends with the guarantee of comprehensively measuring results. From a client's viewpoint it is obvious why calibration data are necessary and how these data are to be processed.

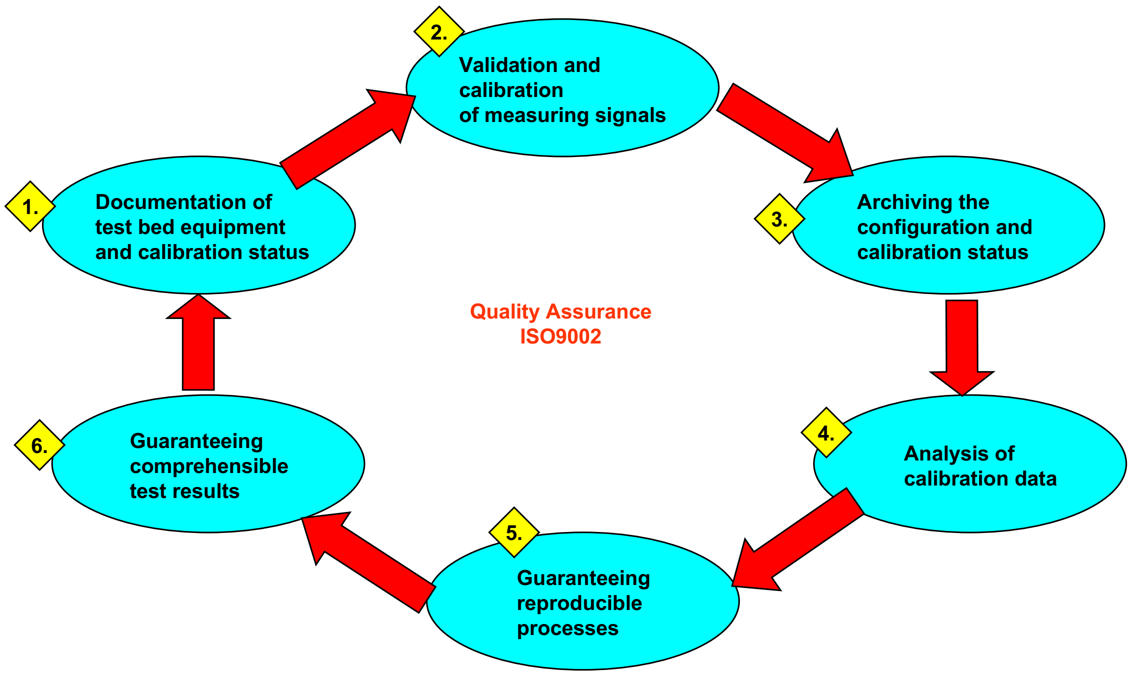

2. Validation and calibration of measuring signals
3. Archiving the configuration and calibration status
4. Analysis of calibration data
5. Guaranteeing reproducible processes
6. Guaranteeing comprehensible test results
1. Documentation of test bed equipment and calibration status
Quality Assurance ISO9002

Figure 1: Process Cycle for continued Quality Verification and Certification

This chapter describes how the calibration service is administratively embedded in a test field. After that the calibration process is described in general. Section 12.4 defines in consecutive order the application elements for the data model and explains the data model as a whole.

12.2 CALIBRATION DATA IN THE TEST FIELD CONTEXT

This section describes a general administration process to handle test bed related calibration data inside an ASAM ODS server from a client's viewpoint.

12.2.1 THE IMPACT OF THE MEASURING EQUIPMENT ADMINISTRATION

A general view on the calibration process leads to the insight that a calibration process is not a stand-alone process but is embedded in the overall measuring equipment administration (figure 2).

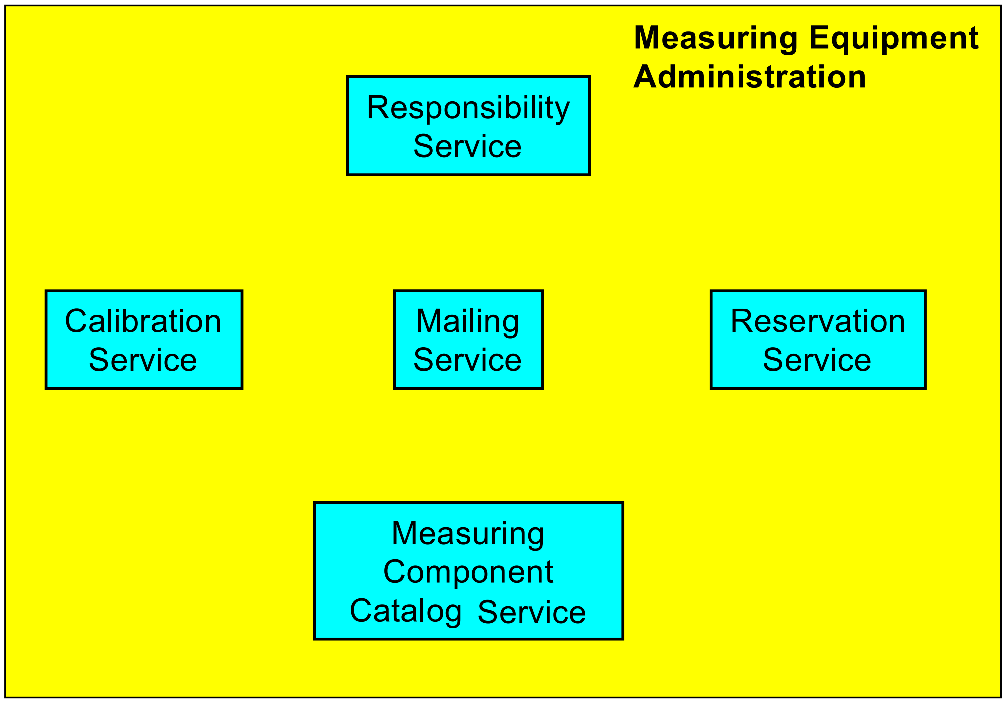


Figure 2: The Calibration Process as an Embedded Service

The measuring equipment administration is composed of five services:

- Calibration Service (this document)
- Reservation Service
- Catalog Service
- Responsibility Service
- Mailing Service.

The services are all related with each other to gain additional values. Example: the Calibration Service detects a measuring component that is out of order. The Calibration Service informs all engineers who had used it. The information process relies on the

- Reservation Service to retrieve the users of the malfunctioning component,
- Mailing Service to send messages.

The purposes of the other services are:

- The Responsibility Service documents which person from the calibration staff is responsible for the calibration of specific measuring equipment. The responsibility relation between a measuring component and a person may change over time. Any relation changes are documented.
- The Mailing Service is used to send messages to registered persons. The service is used by the other Administration services.
- The Catalog Service can be compared with a dictionary summing up all available measuring components, their capability and characteristics. The catalog service is

typically used by the Reservation Service to retrieve and identify an appropriate component according to a requested profile.

12.2.2 THE IMPACT OF THE CALIBRATION LOCATION

Depending on the overall pursued company strategy the calibration process is executed at different locations:

- ➢ test stand,
- ➢ service laboratory, and
- ➢ calibration laboratory.

Depending on the required calibration quality, or other constraints (e.g. removeability of a component from test stand), the calibration process is executed at the appropriate location. Hence different calibration locations may be in operation in parallel at the same company. The only requirement is, that the calibration tools operate on the same data base.

The structure of a test stand and its instrumentation with measuring components may be fixed or is totally dynamic. The latter case is given, if the calibration process is no longer executed at the test stand but at a service or calibration laboratory. In this situation the laboratory provides a stock with calibrated components. When a test stand needs a set of calibrated measuring components, the components are taken from the stock and returned after usage (under involvement of the Reservation Service). By this the components are rotating throughout the test field. The calibration service cares about the status of all returned components inside the stock. In case a calibration check is necessary the components rotate between the stock and the laboratory (figure 3).

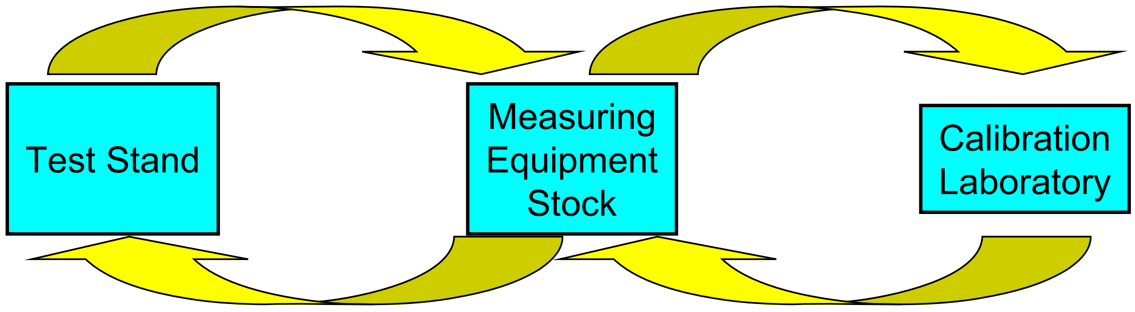


Figure 3: Measuring Components rotating between Test Stand, Stock and Laboratory

As no fixed relation exists between a test stand and the used measuring components it is not convenient to mix the calibration data with test results from any test run on a test stand. The calibration data server should be kept at a separate ASAM ODS Server archiving only the calibration data of all measuring components throughout the test field (figure 4).

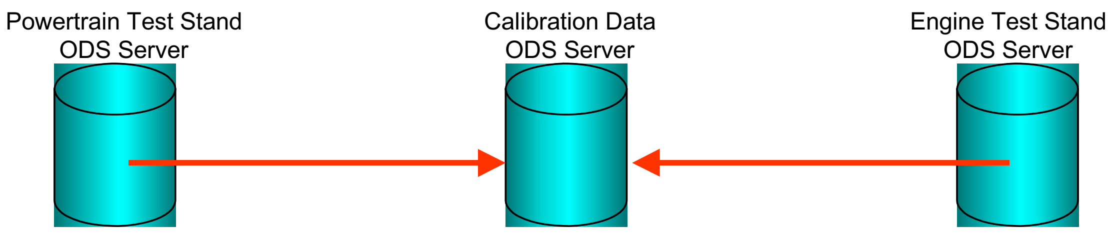


Figure 4: Test Stand ODS Servers linked to the Calibration Data ODS Server

The separation of the calibration data from test stand measuring data on a separate ODS server simplifies the internal application structure of the various test stand ODS Servers. Every measuring component in a test stand's ODS server holds an additional reference to the actual calibration documentation inside the calibration ODS server.

12.2.3 THE IMPACT OF THE CURRENT ODS-BASE MODEL

A general view on a calibration process and its mapping to the existing ODS-Base Model leads to the following insight:

The calibration process of measuring components is regarded as a normal test run. This assumption supports the following mappings:

a) The ODS-section "Administrative Data" holds a calibration project(s) with the various re-calibration tests.

b) The ODS-section "Descriptive Data" holds the set of all measuring equipment components available in the test field. The set of test sequences are the programs needed for the execution of the various calibration processes.

c) The ODS-section "Measuring Data" holds the measured samples collected during the calibration process for each specific measuring component.

d) The ODS-section "Quantities and Units" holds the process quantities and the physical dimensions that are associated with the measured samples.

e) The ODS-section "Security" controls access to the calibration data.

Organizing the calibration data as described above leads to the essential benefit:

➢ the current ODS-Base Model is capable to directly support the requested calibration structure.

Furthermore a trend analysis can be done on the calibration results of each measuring component from the various re-calibration runs.

12.3 THE CALIBRATION PROCESS IN GENERAL

This section deals with the calibration process itself and the resulting concept for storing calibration data inside an ODS-server.

The calibration service uses tools that operate independently from the execution location. The calibration tool can be used at a laboratory as well as on a test stand and accesses the ASAM ODS server with the calibration data. In both cases a unified calibration process is guaranteed. Figure 5 displays the Calibration Service and its internal data flow.

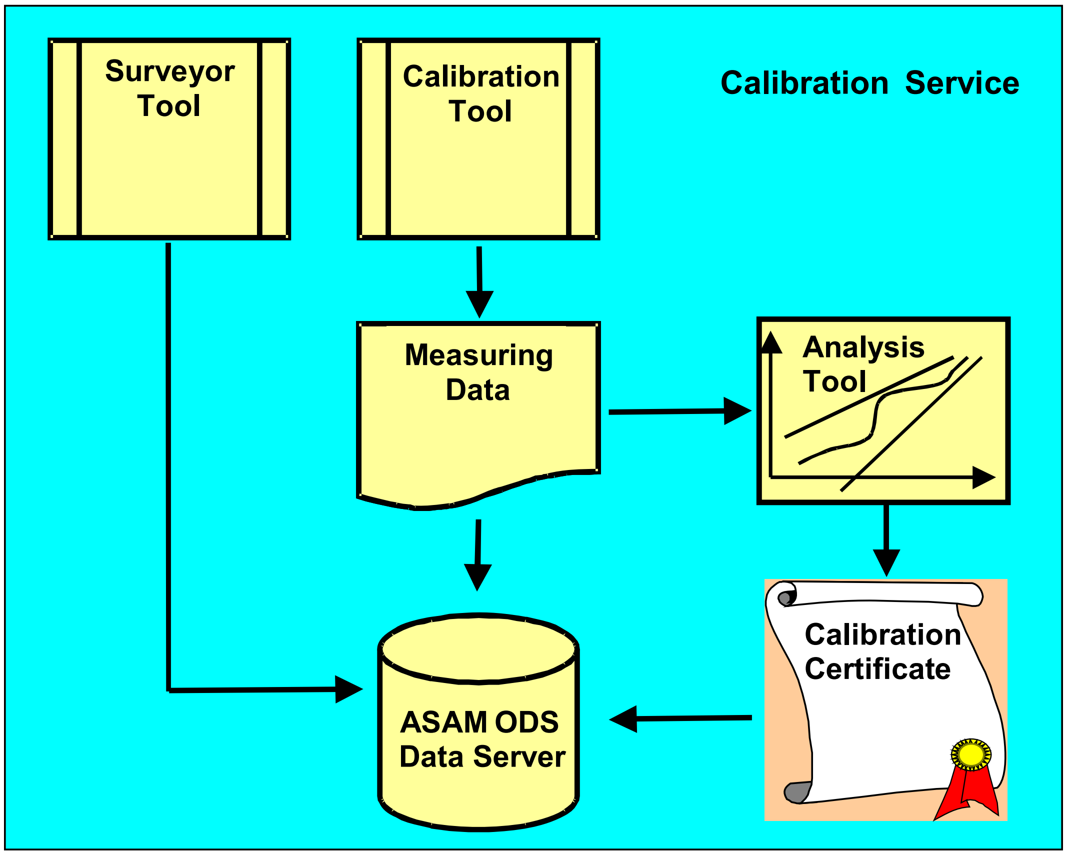


Figure 5: General Data Flow of the Calibration Service

The surveyor tool examines the calibration expiry date of all components, and generates a list of components that need a re-calibration.

- The calibration tool is responsible to carry out the calibration process itself. It triggers the measurement component under test, and in parallel samples measuring data, that are archived in the ODS server. The measuring data is also the input for an analysis tool to validate whether the inspected component is compliant with the calibration specification. In the positive case the analysis tool automatically generates the calibration certificate, which summarizes the substantial results from the calibration process. The calibration certificate is printed out and stored additionally in secured form (e.g. as PDF file) in a data-base. The additional archiving of the calibration certificate in secured form is done, because the measuring data it relies on, can theoretically be changed subsequently manually.

12.4 THE CALIBRATION APPLICATION MODEL

Figure 6 presents the subset of the ODS Base Model that is relevant for the calibration application model with those highlighted Base Elements from which specific application elements are derived to build the calibration application model. The following sections describe in detail the application elements and their attributes. Data types of the attributes are given by the corresponding enumeration name of the DataType enumeration (see section 2.5. for details on data types and their enumerations). Furthermore examples are given for instance elements that are created out of the application elements.

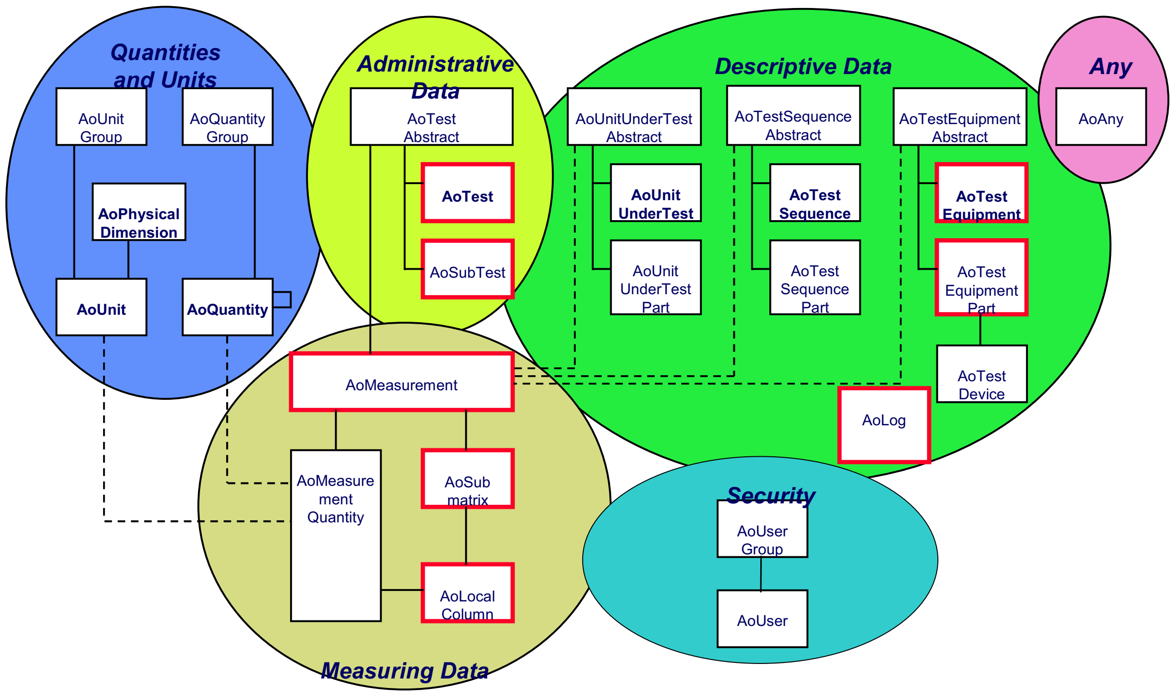


Figure 6: The Base Model with highlighted Base Elements relevant in the Application Model

12.4.1 ADMINISTRATIVE APPLICATION ELEMENTS

All calibrations are administered in projects and subprojects. The component categories are on the topmost project level (e.g. temperature sensors). A subproject aggregates all components belonging to the same category, and a component itself aggregates the result data of all re-calibrations executed so far. Figure 7a gives an example.

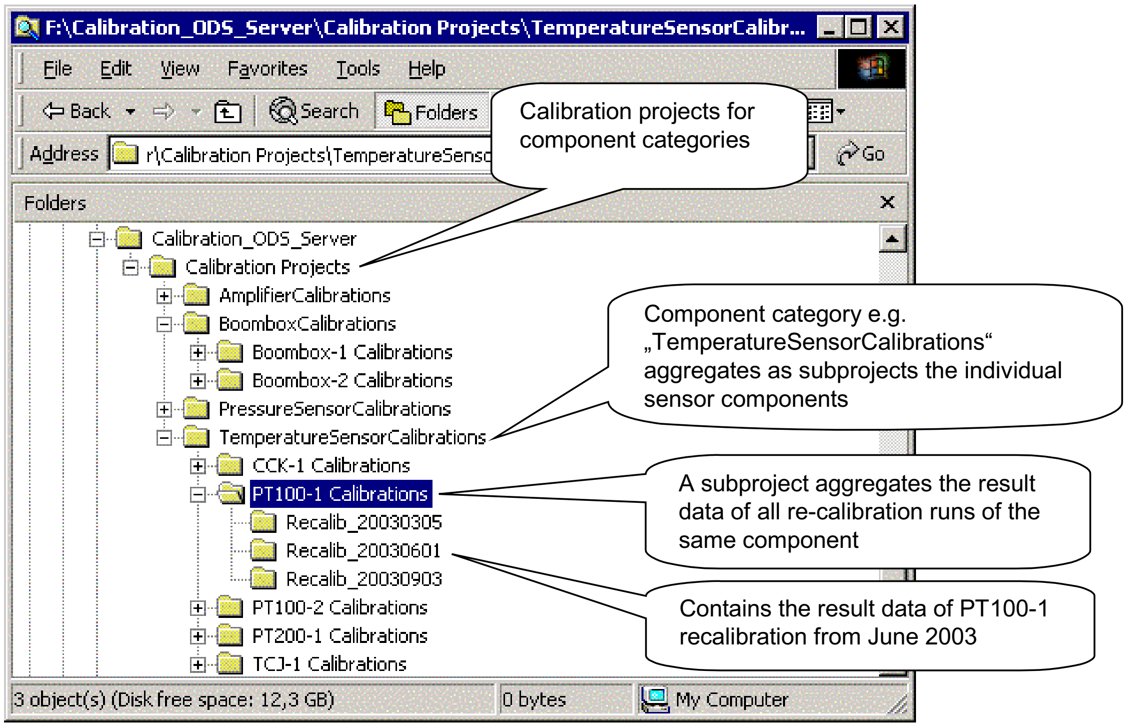

F:\Calibration_ODS_Server\Calibration Projects\TemperatureSensorCalibr...
File Edit View Favorites Tools Help
Back Search Folders
Address r\Calibration Projects\TemperatureSenso
Go
Folders
Calibration_ODS_Server
Calibration Projects
AmplifierCalibrations
BoomboxCalibrations
Boombox-1 Calibrations
Boombox-2 Calibrations
PressureSensorCalibrations
TemperatureSensorCalibrations
CCK-1 Calibrations
PT100-1 Calibrations
Recalib_20030305
Recalib_20030601
Recalib_20030903
PT100-2 Calibrations
PT200-1 Calibrations
TCJ-1 Calibrations
3 object(s) (Disk free space: 12,3 GB)
0 bytes
My Computer
Calibration projects for component categories
Component category e.g. „TemperatureSensorCalibrations“ aggregates as subprojects the individual sensor components
A subproject aggregates the result data of all re-calibration runs of the same component
Contains the result data of PT100-1 recalibration from June 2003

Figure 7a: Administering Calibrations into Projects and Subprojects

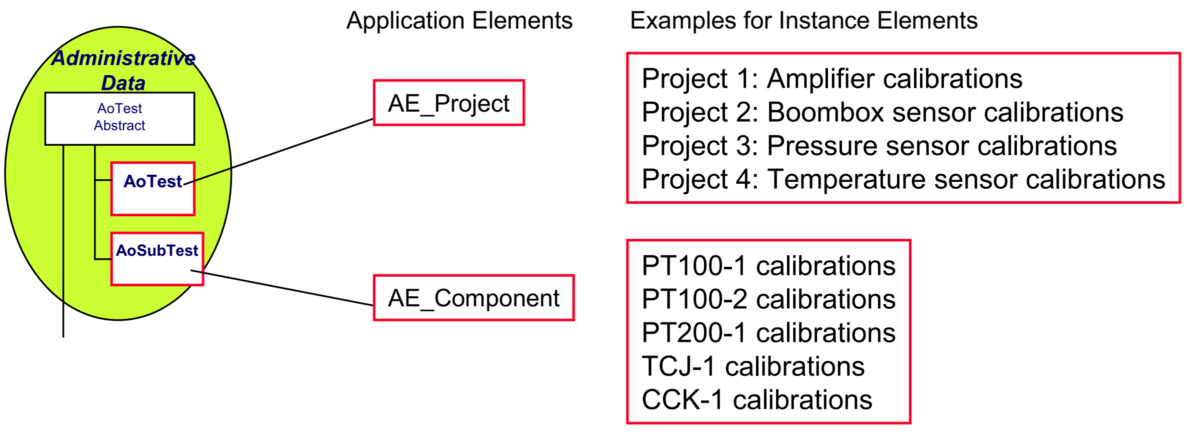


Figure 7b: Mapping the Calibration Administration to ASAM ODS Terminology

The elements AoTest and AoSubTest from the ASAM ODS base model, as shown in figure 7b serve the purpose to derive the calibration specific administrative elements to map the project structure from figure 7a. An element derived from a base element is in the ASAM ODS terminology an “application element”. In the case of AoTest the derived application element is given the name AE_Project. The instances of AE_Project shall have names that denote the component categories for projects like

- AmplifierCalibrations
- BoomboxCalibrations
- PressureSensorCalibrations
- TemperatureSensorsCalibrations
- …

From AoSubTest the application element AE_Component is derived. It serves the purpose to structure the calibrations of a specific component in subprojects. The instances of AE_Component shall have names that denote the individual components, e.g.

- PT100 1
- PT100 2
- TCJ 1
- CCK 1
- …

No specific attributes are defined for the application elements AE_Project and AE_Component.

From AoLog the application element AE_CalibLogbook is derived. The instances of AE_CalibLogbook serve the purpose to document all events around administration and calibration of components. A logbook entry is composed of a set of mandatory attributes and attribute values according to figure 8.

The date attribute value is according to the ASAM ODS convention.

The severity attribute value describes the importance of the logbook entry. The description attribute value is the explanation for the entry. The source attribute value is the origin of the event report. The param attribute value is any data; it can also be a component's name that was calibrated. The user attribute value is the name of the operator reporting the calibration event.

AE_CalibLogbook	**DataType enum name**	**Example, Annotation**
date	DT_DATE	20020401120000000
severity	DT_STRING	info
description	DT_STRING	update accuracy class
source	DT_STRING	calibration tool
param	DT_STRING	torque 650 Nm
user	DT_STRING	administrator

Figure 8: Additional application attributes for Application Element AE_CalibLogbook

12.4.2 MEASURING DATA APPLICATION ELEMENTS

The results of one calibration run for a specific component are organized in a calibration test. The calibration test belongs to a subproject as defined by an instance of AE_Component (figure 7a). Figure 9 gives an example.

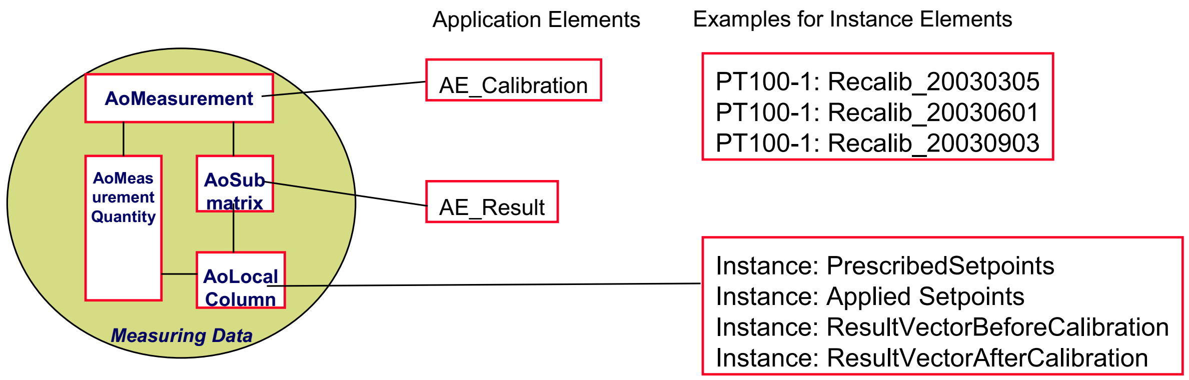

Application Elements
Examples for Instance Elements
AoMeasurement
AoMeasurement Quantity
AoSubmatrix
AoLocal Column
Measuring Data
AE_Calibration
AE_Result
PT100-1: Recalib_20030305
PT100-1: Recalib_20030601
PT100-1: Recalib_20030903
Instance: PrescribedSetpoints
Instance: Applied Setpoints
Instance: ResultVectorBeforeCalibration
Instance: ResultVectorAfterCalibration

Figure 9: Organization of the Results of a Calibration Run.

From AoMeasurement the application element AE_Calibration is derived. The instances of AE_Calibration have the mandatory name attribute (inherited from the base element AoMeasurement) whose value must be unique throughout all calibration runs. The values of the name attribute can be defined like “Recalib_20030305”, “Recalibib_20030601”, … etc., denoting the consecutive re-calibrations by the date.

AE_Calibration	**DataType enum name**	**Example, Annotation**
name (base attribute)	DT_STRING	unique name for calibration run, e.g. calib-1
component (base attribute)	DT_LONGLONG	reference to instance of AE_CalibratedDevice
calibrationGroup (base attribute)	DT_LONGLONG	reference to instance of AE_Component
accuracyClass	DT_LONGLONG	reference to instance of AE_AccuracyClass
calibrationSpec	DT_EXTERNALREFERENCE	reference to applied calibration specification
measurement_begin (base attr)	DT_DATE	calibration begin as utc
measurement_end (base attribute)	DT_DATE	calibration end as utc
operator	DT_STRING	operator’s name executing the calibration
calibrationRangeLower	DT_DOUBLE	calibration range lower value
calibrationRangeUpper	DT_DOUBLE	calibration range upper value

AE_Calibration	DataType enum name	Example, Annotation
processQuantity	DT_LONGLONG	reference to process quantity (AoQuantity)
calibrationNormal	DT_STRING	calibration normal used
calibrationNormalManufacturer	DT_STRING	manufacturer's name of calibration normal
calibrationNormalType	DT_STRING	
calibrationNormalSerialNumber	DT_STRING	serial number calibration normal
calibrationNormalPrecisionClass	DT_STRING	precision class calibration normal
calibrationNormalExpiryDate	DT_STRING	expiry date calibration normal
environmentTemperature	DT_DOUBLE	environment temperature during calibration process
environmentPressure	DT_DOUBLE	environment air pressure
relativeHumidity	DT_DOUBLE	environment air humidity
calibrationCertificate	DT_EXTERNALREFERENCE	external reference to calibration certificate

Figure 10: Attributes of AE_Calibration

From the base element AoLocalColumn four instance elements are derived to document the calibration run. The values for the name's attribute of the instance elements are:

- prescribedSetpoints
- appliedSetpoints
- outputBeforeCalibration
- outputAfterCalibration

The calibration specification prescribes the setpoints and the order of the setpoints to be applied as input to the component under calibration. The actually applied setpoints may differ for any reason. Therefore the actually applied setpoints must be documented.

The behavior of a component under calibration is documented before and after calibration. Therefore the AoLocalColumn instances "outputBeforeCalibration" and "outputAfterCalibration" hold the corresponding output value for each applied input setpoint.

From AoSubmatrix the application element AE_Result is derived. An instance element of AE_Results ties together the AoLocalColumn instances of one calibration run.

12.4.3 DESCRIPTIVE DATA APPLICATION ELEMENTS

All components to be calibrated are collected in a container. The application element AE_DeviceCollection is derived from base element AoTestEquipment.

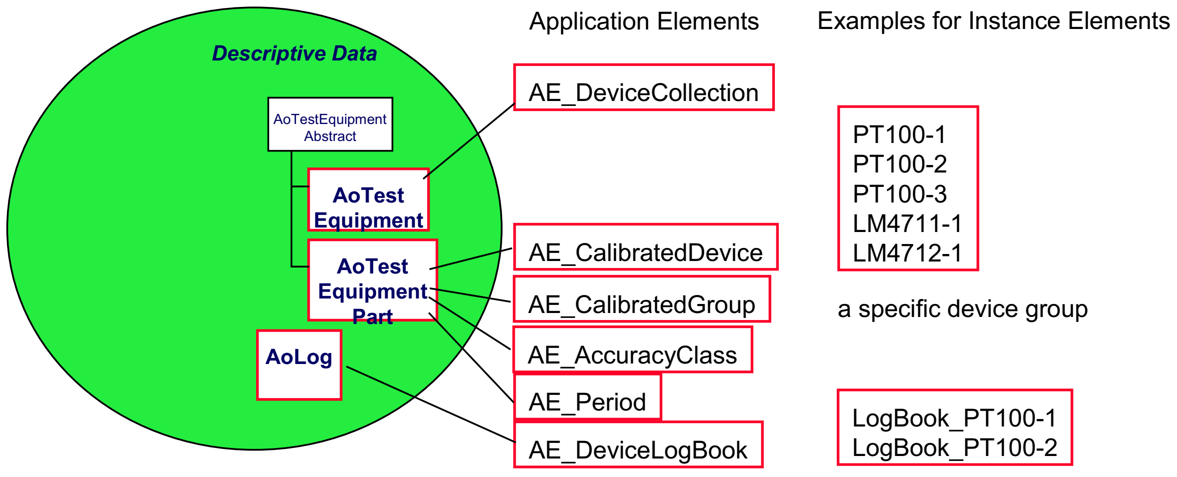


Figure 11: Organization of the Calibrated Components

From AoTestEquipmentPart the application element AE_CalibratedDevice is derived. The characteristics of the instances of AE_CalibratedDevice are documented in detail in figure 12. The list begins with those attributes that identify the device and associate it with its exclusive calibration subproject holding the results from all its re-calibrations.

Several devices may be calibrated as a group and are ordered in the group. The sibling attributes denote previous and following device in the order. The current location and user of the device are also documented. If a device is calibrateable at all is documented by a flag, as is its calibration status (with “is calibrated” or “not calibrated”). This information is accompanied with the definition of the accuracy class after which the device was calibrated.

The signal in/out values specify the operating range of the device. The range is usually defined by the device manufacturer. The allowed environmental operating conditions for air temperature, pressure and humidity are also documented.

AE_CalibratedDevice	**DataType enum name**	**Example, Annotation**
name (base attribute)	DT_STRING	PT100-1
deviceType	DT_STRING	temperature sensor
serialNumber	DT_STRING	
manufacturerName	DT_STRING	
deviceLogbook	DT_LONGLONG	reference to history logbook

AE_CalibratedDevice	DataType enum name	Example, Annotation
calibration (base attribute)	DT_LONGLONG	reference to calibration run belonging exclusively to this device
siblingLeft	DT_LONGLONG	only for ordered devices in groups. Reference to previous device
siblingRight	DT_LONGLONG	only for ordered devices in groups. Reference to following device
deviceLocation	DT_STRING	current location of device
deviceOwner		current user of device
isCalibratable	DT_BOOLEAN	indicates if device is calibrateable
status	DT_BOOLEAN	calibration status
accuracyClass	DT_LONGLONG	reference to instance of AE_AccuracyClass relevant for this device
signalInLower	DT_DOUBLE	minimum signal input level
signalInUpper	DT_DOUBLE	maximum signal input level
signalOutLower	DT_DOUBLE	minimum signal output level
signalOutUpper	DT_DOUBLE	maximum signal output level
temperatureLower	DT_DOUBLE	lower temperature range for proper device operation
temperatureUpper	DT_DOUBLE	upper temperature range for proper device operation
pressureLower	DT_DOUBLE	lower air pressure range for proper device operation
pressureUpper	DT_DOUBLE	upper air pressure range for proper device operation
humidityLower	DT_DOUBLE	lower air humidity range for proper device operation
humidityUpper	DT_DOUBLE	upper air humidity range for proper device operation
calibrationPeriodClass	DT_LONGLONG	reference to period class for re-calibrations
calibrationExpiryDate	DT_DATE	expiry date for current calibration
calibrationPeriod	DT_LONG	period for current valid calibration
calibrationPeriodUnit	DT_STRING	calibration period, can be in days or even in usages
calibratedRangeLower	DT_DOUBLE	lower calibrated input range
calibratedRangeUpper	DT_DOUBLE	upper calibrated input range

Figure 12: Attributes for a Device under Calibration Control

The individual calibration period and calibration expiry date for a device are derived from the calibration period class. The calibration period is always a number, only the unit defines the semantic of the number. The period can be any duration with the unit days for example. However the unit can also be in “usages”. This is essential with crash test, where an acceleration sensor can only be used for a certain number of times, and after that a re-calibration is necessary. Any time period makes no sense.

The last two attributes document the input range of the device for which it is calibrated. This range may be different from the signal input range as specified by the manufacturer of the device.

Another application element is derived from AoTestEquipmentPart to house a set of devices that altogether belong to a complex device like a boombox or measuring chain. The name for this application element is AE_CalibratedGroup. The attributes for this application element are collected in fig. 13, and are rather similar to the attributes of AE_CalibratedDevice. The difference is in the additional definition of two reference attributes (for the 1st and last device) to link the devices as group.

AE_CalibratedGroup	DataType enum name	Example, Annotation
name (base attribute)	DT_STRING	temp chan cooling water, or boombox-1
deviceType	DT_STRING	
serialNumber	DT_STRING	
manufacturerName	DT_STRING	
deviceLogbook	DT_LONGLONG	reference to history logbook
calibration	DT_LONGLONG	reference to calibration run belonging exclusively to the group (if applicable)
device_first	DT_ LONGLONG	reference to 1st device in the group
device_last	DT_ LONGLONG	reference to last device in the group
deviceLocation	DT_STRING	current location of device
deviceOwner		current user of device
isCalibratable	DT_BOOLEAN	indicates if device is calibrateable
status	DT_BOOLEAN	calibration status
accuracyClass	DT_ LONGLONG	reference to instance of AE_AccuracyClass relevant for this group
signalInLower	DT_DOUBLE	minimum signal input level
signalInUpper	DT_DOUBLE	maximum signal input level
signalOutLower	DT_DOUBLE	minimum signal output level
signalOutUpper	DT_DOUBLE	maximum signal output level
temperatureLower	DT_DOUBLE	lower temperature range for proper device operation
temperatureUpper	DT_DOUBLE	upper temperature range for proper device operation

AE_CalibratedGroup	DataType enum name	Example, Annotation
pressureLower	DT_DOUBLE	lower air pressure range for proper device operation
pressureUpper	DT_DOUBLE	upper air pressure range for proper device operation
humidityLower	DT_DOUBLE	lower air humidity range for proper device operation
humidityUpper	DT_DOUBLE	upper air humidity range for proper device operation
calibrationPeriodClass	DT_ LONGLONG	reference to period class for re-calibrations
calibrationExpiryDate	DT_DATE	expiry date for current calibration
calibrationPeriod	DT_LONG	period for current valid calibration
calibrationPeriodUnit	DT_STRING	calibration period, can be in days or even in usages
calibratedRangeLower	DT_DOUBLE	lower calibrated input range
calibratedRangeUpper	DT_DOUBLE	upper calibrated input range

Figure 13: Attributes for AE_CalibratedGroup

The application element AE_AccuracyClass is derived from AoTestEquipmentPart. Instances of AE_CalibratedDevice and AE_Calibration refer to instances of AE_AccuracyClass to document the applied accuracy during a re-calibration.

AE_AccuracyClass	DataType enum name	Example, Annotation
physDimension	DT_LONGLONG	reference to phys dimension
unit	DT_ LONGLONG	reference to unit
precision	DT_LONG	amount of fractional digits
rangeEnd	DT_DOUBLE	calibrated range end
absError	DT_DOUBLE	absolute error
relError	DT_DOUBLE	relative error
accuracyClassDescription	DT_EXTERNALREFERENCE	reference to accuracy class description
prescribedSetpoints	DS_DOUBLE	prescribed set of setpoints
calibrateSetpoint	DS_BOOLEAN	set with flags specifying a setpoint is to calibrate or only to check

Figure 14: Attributes for AE_AccuracyClass

The application element AE_CalibrationPeriod is derived from AoTestEquipmentPart. A period can be a time period or the number of usages. The usual period will be a time period e.g. in days. However in specific applications like crash tests the calibration period is defined by usages. This means, the number of times a device is used, is essential for re-calibration.

AE_CalibrationPeriod	DataType enum name	Example, Annotation
calibPeriod	DT_LONG	time period or usages
periodUnit	DT_ LONGLONG	unit for period

Figure 15: Attributes of AE_CalibrationPeriod

The application element AE_DeviceLogBook is derived from AoLog. Every device under control of the ASAM-ODS calibration server has its own logbook documenting its history.

AE_DeviceLogBook	DataType enum name	Example, Annotation
date	DT_DATE	utc date
operator	DT_STRING	who created the entry
event	DT_STRING	event in short
status	DT_STRING	calibration status
messageText	DT_STRING	explanation

Figure 16: Attributes of AE_DeviceLogBook

12.4.4 THE COMPLETE CALIBRATION DATA MODEL

The complete calibration application model is presented in figure 17 as UML diagram. The yellow boxes are the application elements, and the lines between the boxes are the relations between the application elements. Furthermore besides the yellow boxes the names of the base elements are written from which each application element is derived.

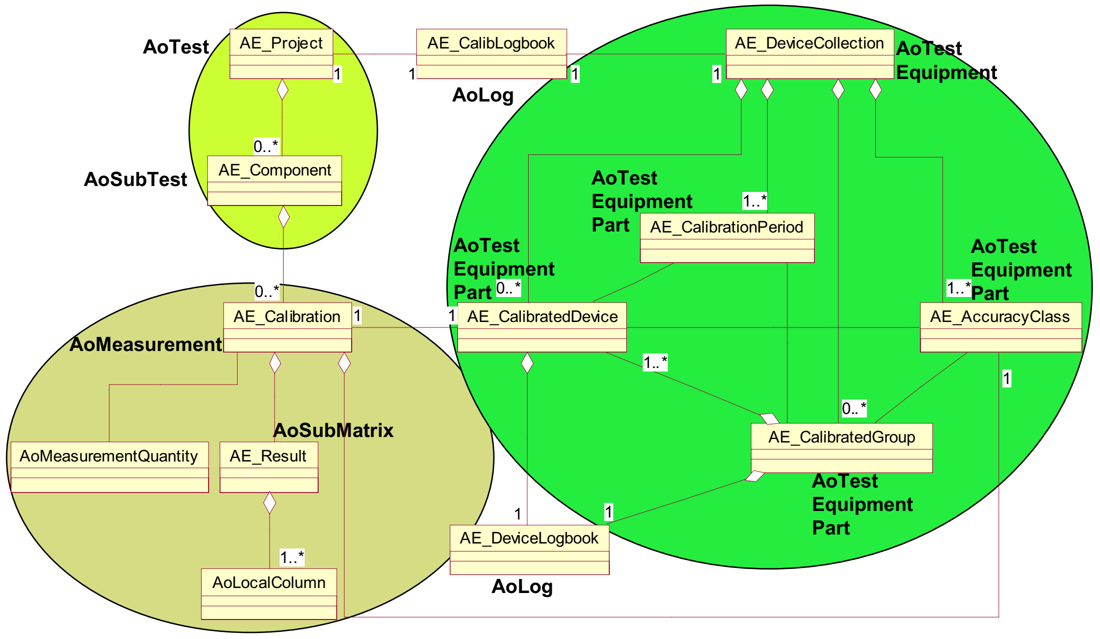


Figure 17: The complete Application Model for Calibration Data

The alternative entry points into the data model are the application elements AE_Project and AE_DeviceCollection. The project entry point leads to the collection of all calibration runs executed so far. The device collection entry point leads to the collection of all devices that are administered by the ODS server. Each instance of a calibrated device is associated with its accuracy class, calibration period, device logbook, and finally with the set of re-calibrations executed so far.

The application element AE_CalibratedGroup is introduced to administer devices that belong to an exclusive component group that must be calibrated in a group. Examples for such groups are:

- a measuring chain that is composed of elements like sensor, amplifier, … etc. .
- a group of independent acceleration sensors that are always used as complete group in crash tests.
- A boombox housing a set of independent measuring components

The application element AE_DeviceLogBook is associated with a calibrated device or a calibrated group. The application element serves the purpose to document a device's resp. a group's history.

The application elements AE_AccuracyClass and AE_CalibrationPeriod are associated with calibrated devices resp. calibrated groups. The instances of these elements serve the purpose to define data sets that are of global interest. All instances of AE_CalibratedDevice reference their specific accuracy class instance that is relevant for calibration. This avoids the surplus to define all details about the applied accuracy class in each calibrated device. This also holds for the calibration period.

The instances of the application element AE_Component are subprojects. Each subproject contains all calibration runs that belong to one specific component, resp. device. The results of each calibration run are united by an instance of AE_Calibration.

A calibration has a relationship to the calibrated device. This organization enables a client application to introspect in an easy way all re-calibrations of a specific device.

The application element AE_Result ties together the AoLocalColumns. Four instances of AoLocalColumn exist for each calibration run:
a) the set of setpoints to apply in a calibration run according to the calibration specification,
b) the set with the really applied setpoints,
c) the behaviour of the device under calibration by the applied setpoints before calibration,
d) the behaviour of the device under calibration by the applied setpoints after calibration.

The instance of the application element AE_CalibLogbook serves as global logbook to document all calibration activities.

12.5 REVISION HISTORY

Date **Editor**	**Changes**
2003-09-29 B. Thelen	Created document
2003-10-14 B. Thelen	Updated section 4.1
2003-10-15 R. Bartz	Some errors have been fixed
2003-12 R. Bartz	Figure 17 has been exchanged by a new one Minor textual changes have been made
2003-12-30 R. Bartz	The **Release** version has been created

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

ICS 01.120

Price based on 537 pages