
HighQSoft



ODS Object Oriented API

Version 1.23

Interface Implementation

Karst Schaap 2007/06/21

Contents

1	General information	1
1.1	Introduction	1
1.2	Pools and maps in the implementation	4
1.3	Performance of the implementation.	6
1.4	There is no return.	9
1.5	Return is a single value or object.	10
1.6	Return is an array	10
1.7	Return is an iterator	10
1.8	Handling of the values of the AoLocalcolumn.	10
1.9	Usage of the value flags.	11
1.10	Logging of the server behaviour	12
1.11	Connect to the ODS API using HighQSoft implementation.	12
1.12	AoServiceFactory	12
1.13	AoService	13
1.14	Transaction handling in ODS API.	14
2	Explanation of the documentaion	17
2.1	Explataion of the method documetation	17
3	Connect to an ASAM ODS server using the implementation of HighQSoft	19
3.1	Connect to the ODS API using HighQSoft implementation.	19
3.2	AoServiceFactory	21
3.3	AoService	21
4	ASAM ODS Interface	23
4.1	AoFactory	23
4.2	AoSession	32
4.3	ApplElemAccess	58
4.4	ApplicationAttribute	98

4.5	ApplicationElement	117
4.6	ApplicationRelation	141
4.7	ApplicationStructure	155
4.8	BaseAttribute	174
4.9	BaseElement	177
4.10	BaseRelation	181
4.11	BaseStructure	185
4.12	Blob	192
4.13	Column	196
4.14	ElemResultSetExtSeqIterator	203
4.15	EnumerationDefinition	205
4.16	InstanceElement	209
4.17	InstanceElementIterator	234
4.18	Measurement	237
4.19	NameIterator	240
4.20	NameValueIterator	243
4.21	NameValueUnitIdIterator	245
4.22	NameValueUnitIterator	248
4.23	NameValueUnitSequenceIterator	252
4.24	Query	254
4.25	QueryEvaluator	258
4.26	SMatLink	261
4.27	SubMatrix	268
4.28	ValueMatrix	270
5	ASAM ODS Structures	297
5.1	Definitions of the ACL structure.	297
5.2	Definitions of the AIDName structure.	297
5.3	Definitions of the AIDNameUnitId structure.	297
5.4	Definitions of the AIDNameValueSeqUnitId structure.	298
5.5	Definitions of the AIDNameValueUnitId structure.	298
5.6	Definitions of the ApplAttr structure.	299
5.7	Definitions of the ApplElem structure.	300
5.8	Definitions of the ApplicationRelationInstanceElementSeq structure.	301
5.9	Definitions of the ApplicationStructureValue structure.	301
5.10	Definitions of the ApplRel structure.	301
5.11	Definitions of the AttrResultSet structure.	303

5.12	Definitions of the ElemId structure.	303
5.13	Definitions of the ElemResultSet structure.	303
5.14	Definitions of the ElemResultSetExt structure.	304
5.15	Definitions of the InitialRight structure.	304
5.16	Definitions of the JoinDef structure.	305
5.17	Definitions of the NameUnit structure.	305
5.18	Definitions of the NameValue structure.	305
5.19	Definitions of the NameValueSeqUnit structure.	306
5.20	Definitions of the NameValueSeqUnitId structure.	306
5.21	Definitions of the NameValueUnit structure.	307
5.22	Definitions of the NameValueUnitId structure.	307
5.23	Definitions of the QueryStructure structure.	307
5.24	Definitions of the QueryStructureExt structure.	309
5.25	Definitions of the RelationRange structure.	310
5.26	Definitions of the ResultSetExt structure.	310
5.27	Definitions of the SelAIDNameUnitId structure.	310
5.28	Definitions of the SelOrder structure.	311
5.29	Definitions of the SelValue structure.	311
5.30	Definitions of the SelValueExt structure.	311
5.31	Definitions of the T_COMPLEX structure.	312
5.32	Definitions of the T_DCOMPLEX structure.	312
5.33	Definitions of the T_ExternalReference structure.	313
5.34	Definitions of the T_LONGLONG structure.	313
5.35	Definitions of the TS_Value structure.	313
5.36	Definitions of the TS_ValueSeq structure.	314
6	ASAM ODS Unions	315
6.1	Definitions of the SelItem union.	315
6.2	Definitions of the TS_Union union.	315
6.3	Definitions of the TS_UnionSeq union.	317
7	ASAM ODS Enumerations	321
7.1	Definitions of the AggrFunc enumeration.	321
7.2	Definitions of the AttrType enumeration.	321
7.3	Definitions of the BuildUpFunction enumeration.	322
7.4	Definitions of the DataType enumeration.	322
7.5	Definitions of the ErrorCode enumeration.	325

7.6	Definitions of the JoinType enumeration.	330
7.7	Definitions of the QueryStatus enumeration.	331
7.8	Definitions of the Relationship enumeration.	331
7.9	Definitions of the RelationType enumeration.	332
7.10	Definitions of the RightsSet enumeration.	332
7.11	Definitions of the SelOpcode enumeration.	332
7.12	Definitions of the SelOperator enumeration.	334
7.13	Definitions of the SelType enumeration.	334
7.14	Definitions of the SetType enumeration.	334
7.15	Definitions of the SeverityFlag enumeration.	335
8	ASAM ODS Constants	337
8.1	Definitions of the LockMode constant.	337
8.2	Definitions of the QueryConstants constant.	337
8.3	Definitions of the ResultType constant.	338
8.4	Definitions of the SecurityLevel constant.	338
8.5	Definitions of the SecurityRights constant.	339
9	ASAM ODS Types	341
9.1	Definition of the CORBA types.	341
9.2	Definitions of the Scalar type.	341
9.3	Definitions of the Sequence type.	343
10	ASAM ODS Exceptions	353
10.1	AoException	353
11	ASAM ODS Interface Short Reference	355

Chapter 1

General information

1.1 Introduction

The ASAM ODS OO API is defined in the Corba Interface Definition Language. The interface can be used as a Client/Server interface. The server part is implemented in the Avalon. The client part can be generated by an IDL compiler. Most corba implementations have their own IDL-Compiler to generate the client stub. For other languages the same procedure must be done with just another IDL compiler and language compiler.

Figure 1.1: IDL-Compiler

The interfaces defined in the ASAM ODS API are in the Corba objects which will stay on the server and the access will be done by methods. On the client there is only a proxy object available so each access will cause a network round trip. These network round trips can lead to bad performance. It is required to get a good balance between the access to structures and to the interfaces. The access to the interfaces is easier to program but as soon as a large number of instances will be accessed the performance will be too low.

The number of objects accessed by the client in one session also has an influence on the performance of the server, because the server has to store the object reference in a map. Each time a new object is created and the access to one object the server will search in the map for the object, the performance will reduce when there are more objects stored in the map.

Since ASAM ODS 5.0 there are destroy-methods defined in the interface, so the client can tell the server to remove an object from the map. It is the responsibility of the client to call these destroy-methods. The server can destroy the objects, except when the client calls the destroy-method, at the close of the session.

The AoFactory is published at the Corba-NameService, the client gets the proxy object from the Corba-NameService and from this object with the interface methods of AoFactory, the client can connect to the server using the methods newSession. From the session, the client can get the application structure or the base structure and so on. The methods of the interface are so designed that from the most interfaces there is a way back to the session, an enterprise client doesn't need to transport the connect information, if only an instance element is required at another component of the client software, which can make the client more reliable for username and password protection.

Athos offers some client-side classes, AoServiceFactory and AoService, which hide the access to the

Corba-NameService. The classes are documented in the Framework-documentation and are no part of this document.

Athos offers the clientside classes for the direct access, these classes don't use Corba and the performance difference can be significant.

The main interfaces of the ASAM ODS API are given below:

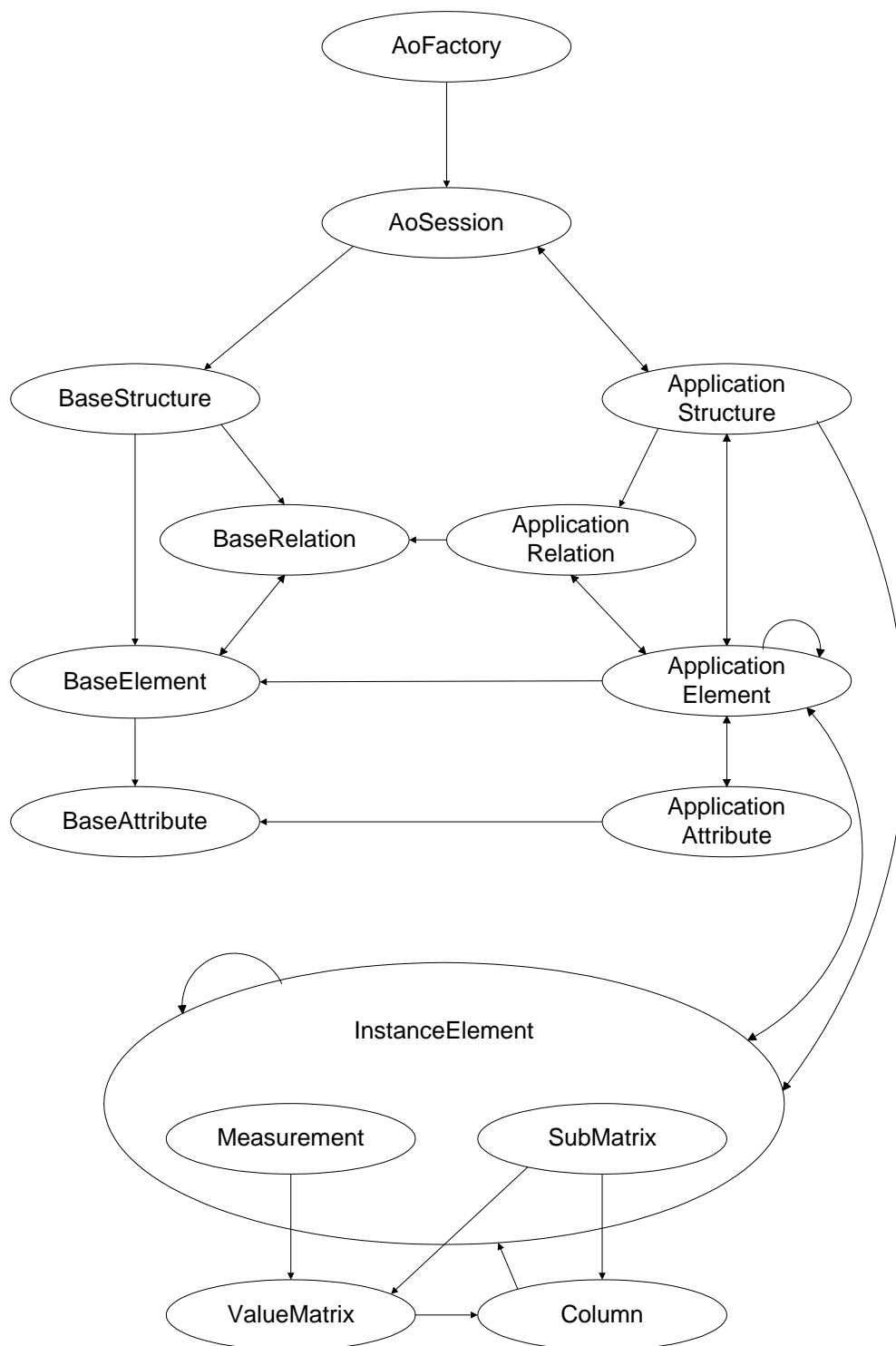


Figure 1.2: The main components

1.2 Pools and maps in the implementation

The pools and maps in the implementation are important for the performance and the actuality of the instance data. The pools and maps can be controlled by some INI-file variables and depends on the used implementation. The following figure shows the pools and maps in the implementation.

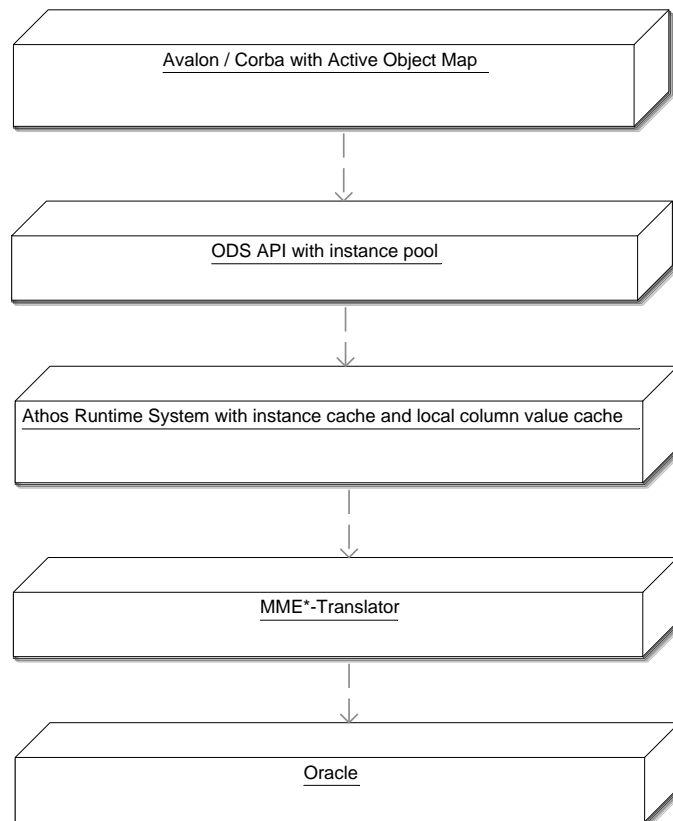


Figure 1.3: The pools and maps in the implementation

When the number of objects in a pool or map increase the performance of the system will be less because of search in the pool or map takes time. The balance between the number of objects in a pool and the usage of the pool depends on the application.

1.2.1 Avalon / Corba with Active Object Map

The corba implementation of the avalon uses according the definition in ASAM ODS the Portable Object Adapter (POA) of the objects. POA is the implementation independent transfer of the objects between different corba implementations. The POA objects are all stored in an active

object map, all access from a client to an object on the server the corba implemetation at the server search for the object in the active object map. The more objects in the active object map the less the throuput of the client / server communication. To prefend influence of the number of objects in the active object map of other clients for each session (AoSession) an own active object map is created. To reduce the numer of objects in the active objects the ODS API have defined the `destroy()` method at different instances like `InstanceElement`. The `destroy()` method only removes the object from the active object map, the destroy method have no influence on the persistence of the object in the server.

1.2.2 ODS API with instance pool

The instance pool in the ODS API have a little influence on the size of the active object map of Avalon. When the instance pool is activated the instances with from the same instance in the athos runtime system becomes the same entry in the active object map. An instance pool is created at each application element so there are less instances in the instances pool as there are objects in the active object map.

The INI-file variable `ODSAPI_USE_INSTANCE_POOL` switch the usage of the instance pool in the ods api on or off at startup of the implementation. For a detail description of the INI-file variables look in the Technical Reference Sheet of the ASAM-ODS API.

1.2.3 Athos Runtime System with instance cache and local column value cache

At the Athos Runtime System there is a cache for instances and for the values of the local column. The cache for the instances is important for the performance of the translator / server. Translators with a database and fast database access should no use the cache of the instances in the Athos Runtime System, however translators with slow access to the instances (reading files / schanning directories) can get a lot of performance when the cache of the instances is used.

With the INI-file variable `NO_INSTANCE_CACHE` can the instance cache be deactivated. When the cache of the instances is not active the pool with instances will be removed before the next access time the instances are loaded from the translator into the Athos Runtime System. The cache of the instances use the Id as index, all access to the instance with the Id will not cause that the instances are removed from the cache. The instances are not removed from the cache when a session (AoSession) is closed except when the INI-file variable `CLEAR_INSTANCE_CACHE` (available since Athos 3.5c) is set.

The INI-file variable `CHECK_ID_ALWAYS` for the implementation for each new instance to check the Id of the instance element, so that no two instances with the same Id are stored in the pool. The implementation knows either there are instances already in the pool or the pool is empty before the instances are loaded from the data storage. When no instance cache is used (`NO_INSTANCE_CACHE=YES`), the pool will be often empty and the there are no instances with the same Id in the storage then the Ccheck of the Id is not required.

The cache of the values of the local column is used for faster access to the values of the localcolumn, normally the values of the same local column are accessed by the client more then once. This cache is also the maximum of the size of one value of the local column. The cache can be corntrolled by the following INI-file variables `MAX_LC_MEMORY`, `MAX_NUMBER_LC` and `FREE_NUMBER_LC`. The INI-file variable `MAX_LC_MEMORY` defines the number of memory in MBytes, the INI-file variable `MAX_NUMBER_LC` defines the number of local columns in the cache, the INI-file variable `FREE_NUMBER_LC` defines the number of local columns removed from the cache when there are to much local columns in the cache. These two last INI-file variables controls the local column value cache when the are a lot of local columns with only a few number of values. The cache is cleared on

close of the session (AoSession) when the INI-file variable `CLEAR_LOCALCOLUMN_MEMORY` is set. The cache is turned off when the INI-file variable `MAX_LC_MEMORY` is set to 0. When there are a lot of local columns with only a few values the cache of the values of the localcolumn is not required, when the localcolumns are removed from the instance cache. For a detail description of the INI-file variables look in the Technical Reference Sheet of the ASAM-ODS API or the Athos Runtime System.

1.3 Performance of the implementation.

In the section **Pools and maps in the implementation**(p. 4) the usage of the different pools and maps is explained, besides the pool there are a lot of other influences on the performance of the system. The anti-virus is one of the main performance killer in a multi computer environment. Also the application model have influence on the performance of the system.

1.3.1 ASAM ODS and anti-virus programs

The anti-virus programs have influence on different places in an ASAM ODS implementation. The following figure shows the system components, the anti-virus program can have influence on each level.

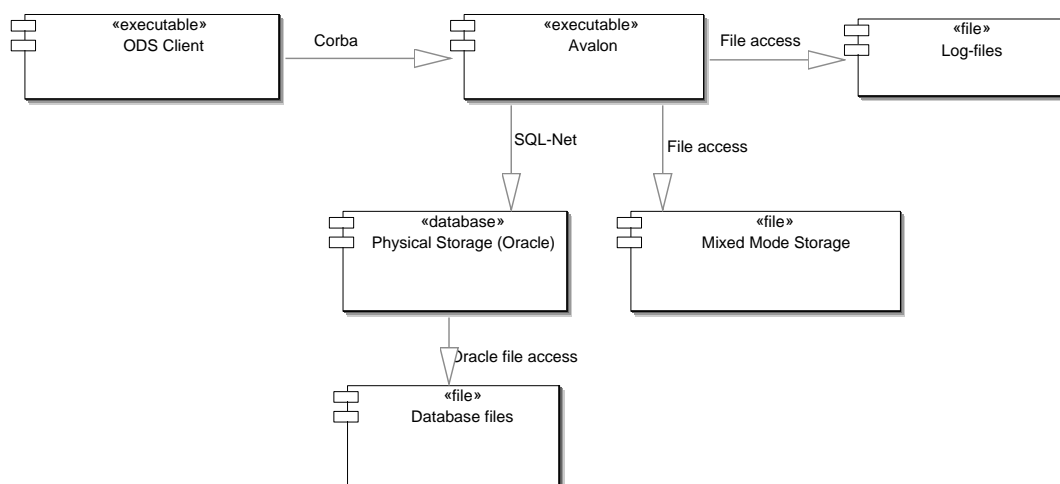


Figure 1.4: The system components

The corba communication between the ODS Client and the Avalon.

The anti-virus program normally checks the incoming packages from the network. All the communication between the ODS Client and the Avalon is checked either at the Avalon side

or on the ODS Client side. At each network round trip, call from client to server, the anti-virus will run first on the Avalon side to check the incoming request and then at the ODS Client side to check the return of the request.

The file access to the LOG-files.

The anti-virus program normally checks the writing to the files system. The LOG-files of the Avalon are stored on the file system, so the anti-virus will run every time the Avalon write something in the LOG-files. Avalon flushes every line to the LOG-files to guarantee that no lines get lost at the end of Avalon so the anti-virus runs at every line written from the Avalon in the LOG-file. Anti-Virus programs normally scans the complete file, so when the LOG-files are big, the scan of the anti-virus will cost time. This can be controlled by the INI-file variables `DEBUGLEVEL` , `ERR_MAX_LINES` and `ERR_MAX_LINES`. When the LOG-files are located on a network drive the anti-virus run another time during the network transfer.

The file access to the Mixed Mode Storage.

Like the file access to the LOG-files the anti-virus will check the acces to the Mixed Mode files also. The Mixed Mode files are big binary files, so the anti-virus program will have problems to check the files. When the anti-virus checks the files only when they are written files are checked not at every read access so the influence on the performance is only during the write of the Mixed Mode files. Avalon writes the values of one localcolumn in one write access so at every local column the anti-virus will run once. When the Mixed Mode files are located on a network drive the anti-virus run another time during the network transfer.

SQL-Net from the Avalon to the Oracle database.

The SQL-Net is also a network transfer between the Avalon and Oracle so here can be also a run of the anti-virus program. We are no oracle specialist, maybe oracle use not the operating system functionality for the transfer but has it own and the anti-virus will not run during this communication.

Oracle file access

Oracle stores the information also in file an the file system, so here can the anti-virus be also active. We are no oracle specialist, maybe oracle use not the operating system functionality for writing the files and the anti-virus will not run when oracle update there files. The Oracle database files are huge files so when the anti-virus runs it will have big influence on the performance.

1.3.2 The application model and the performance.

The application model have influence on the performance of the implementation. The ASAM ODS physical storage for relational database defines that the attributes of an element is stored in one table, so there is a fast access to these attributes. However there are some datatypes or relations which can not be stored in a single column of the table and stored in a separate table or special column. The application model designer should try to avoid these datatypes because the reduce the performance of the implementation.

N:M-relations

N:M-relations are stored in a separate table. Access to these relations force the implementation to an extra access on the data storage. This access is done for each N:M relation at each instance. It is not possible to load the relation information for multiple instances at once like it is done for the 1:N relations. Modification of a N:M relation force the implementation to load the relation for both instances, so all N:M relations for both instances are loaded. Try to reduce the N:M relations to the static part of the application model like `AoUser` and

AoUserGroup and avoid the N:M relation at the dynamic parts like AoMeasurement or AoMeasurementQuantity. The worst thing according the performance is a N:M relation between the static part and dynamic part of the application model.

Datatype with DT_BLOB and DT_BYTESTR

Attributes with the datatype DT_BLOB and DT_BYTESTR are stored in the oracle datatype 'long raw', to load these oracle datatype from the data storage a special call is required. The implementation need for these datatype an additional call to the data storage for each instance element. As long as the attribute is not requested by the client the attribute is not loaded from the data storage. When the instance is modified the attribute must be loaded from the data storage.

Sequence datatypes or complex datatypes.

Attributes with a sequence datatype or complex datatypes are store in a separate table. When the extended query is used these attributes can be combined with a join and the access to these attributes is only one request together with the other attributes from the storage. The implementation needs to compress the result from the data storage as result for the client. When the instances are not loaded with the extended query the access to these attributes force an additional request from the implementation to the data storage. When the instance is modified the attribute must be loaded from the data storage. A database index on the columns iid and ord can improve the performance.

Instance attributes.

Instance attributes are not part of the application model but attribute missing in the application model. All instance attributes are stored in the same separate table SVCINST. The instances attributes can not be loaded with a query. The implementation need the instance attributes an additional call to the data storage for each instance element. When the attributes are not requested by the client the instance attributes are not loaded, except when the instance element is modified the instances attributes are loaded. Even when no instance attributes are given for the instance as soon as there is one instance attribute for any instance of an application element the implementation checks the table SVCINST when any instance of the element is modified. A database index on the columns aid and iid can improve the performance.

1.3.3 INI-file variables for communication with the data storage.

The performancne fo the system depends on the performance on the communication between the implementation and the data storage (Oracle database). There are some variables which have influence on the behaviour of the impementation.

Oracle sequences for the Id

The specification of the ASAM ODS Physical Storage requires oracle sequences for the Id. The implementation accepts also a physical storage without these oracle sequence. The usage of the oracle sequences are much faster then calculating the MAX and add one for the next instance element especially for element with a lot of instances. The INI-file variable USE_ID_SEQUENCE controls the usage of the these sequences. With the tool UpdatStorage you can create the sequences.

The number of instances loaded from the database.

The number of instances loaded from the data storage has influence on the performance of the implementation. The number reduce also the amount of memory allocated before the ther equest is done, when a lot of attributes are requested with a high number of instances the implementation tries the allocate memory for each attribute according the datatype for that

number of instances. Allocating this memory can force other memory to be swapped out in the virtual memory which will reduce the performance. This can be controlled by the INI-file variable `SQL_MAX_ROWS` and the `how_many` argument at the query methods of the interface `AppElemAccess`. Besides the total number of instances also the number of instances loaded in one package from the data storage have influence on the performance. The number of instances in one call requested from the storage is controlled by the INI-file variable `ORACLE_HOST_ARRAY_SIZE`. Here also the bigger the number of the host array size is the more memory is required. This works also only on the query methods of the interface `AppElemAccess`.

1.3.4 Oracle performance optimizing.

Optimizing the oracle storage is a special job. We are no oracle specialist and therefore not able to optimize the oracle storage. We give here only a few hints for optimizing the data storage.

Index on the table SVCVAL.

An index on all columns except the column `VALBLOB` increase the read access to the table `SVCVAL` where the values of the local columns are stored. An index reduce the performance during write but the advantage at read access is must better.

Index on columns for relational attributes.

A relation attribute is an attribute at an element which points to another element. The Father/Child relations are always relational attributes at the child element. The relational attributes are used in common for the navigation. A database index on the columns of these relational attributes can increase the performance, especially in the area below the element of type `AoMeasurement`. The relational attribute from `AoMeasurementQuantity` or `AoSubMatrix` to `AoMeasurement` are used to find the values of the local columns. An index reduce the performance during write but increase the performance at read access, so the implementation must be check that the complete performance is increased. When writing instances the implementation have also read access to the data storage.

Analyze of the storage.

Oracle offers the opportunity to analyze the tables or the complete data storage. After an analyze of the data storage the access is normally faster but can also reduce the performance of the system. An analyze of the table `SVCVAL` had always better performance at our systems.

A constraint on the relational attribute from `AoLocalColumn` to `AoMeasurementQuantity`.

When an instance of `AoMeasurement` is deleted the instances of `AoMeasurementQuantity`, `AoSubMatrix`, `AoLocalColumn` and `AoExternalComponent` must be deleted also. The implementation deleted first the instances in the following order `AoExternalComponent`, `AoLocalColumn`, `AoSubMatrix` and `AoMeasurementQuantity`. When an instance of `AoMeasurementQuantity` is deleted, the corresponding instances of `AoLocalColumn` must be deleted also, the implementation have to check this. When a database constraint exist the data storage checks it an gives an error message so the implementation knows that there are instance of `AoLocalColumn` to delete. With the INI-file variable `ORACLE_CONSTRAINT_LC_MEQ` the implementation knows that the constraint exist and don't check the instance of `AoLocalColumn`.

1.4 There is no return.

The methods in the ASAM ODS OO_API which have no return have the same behaviour within the API.

The method throws an exception when an error occurs.

1.5 Return is a single value or object.

The methods in the ASAM ODS OO_API which return a single value have the same behaviour within the API.

The method will throw only an exception if an error occurs. A null-pointer or null-object will be returned if the null-object is a valid return. E.g the Method `getBaseAttribute` of the interface `ApplicationAttribute` will return a null-object if the application attribute is not derived from a base attribute.

1.6 Return is an array

The methods in the ASAM ODS OO_API which return an array have the same behaviour within the API.

The method will always return an array as long as no error is occurred. If there are no elements for the requested method an empty array, `length = 0` will be returned.

1.7 Return is an iterator

The methods which will return an iterator have all the same behavior.

An iterator will be returned always as long as there happens no error in the methods. If there are no entries available in the iterator, the iterator will be returned by the method.

All iterators have the following methods

- `getCount()`, this method will return the number of elements which by the iterator.
- `nextOne()`, will return the next element of the iterator, if the last element is reached the method returns a null-element. If there are no elements in the iterator at all the first call of the method `nextOne()` will return also null-object. See also **Return is a single value or object.** (p. 10)
- `nextN()`, will return an array of elements. If there are less elements left in the iterator as requested by the client, the array will be filled with the available number of elements. If there are no elements left the returned array will be an empty array, `length of array = 0`. See also **Return is an array** (p. 10)
- `reset()`, will reset the 'current element pointer' to the beginning of the of the iterator.
- `destroy()`, will destroy the iterator-objects on the server, any access to the iterator after the destroy method will lead to an error. The objects, returned with the methods `nextOne()` and `nextN()` returned to the client will stay active and will not be destroyed by this method.

1.8 Handling of the values of the AoLocalcolumn.

In this section when we talk about the values of the `AoLocalColumn` we mean the values of the attribute derived from the base attribute values at an instance element from an application element of `AoLocalColumn`. The same naming is used for the other base attributes. We use the names from the base model but at the client programmer should be aware that the names of the application attributes and application elements can differ from the names of the base model.

The values of the AoLocalColumn can be accessed on two different ways:

1. Using the attribute methods of the interface **InstanceElement**(p. 209). With these methods such as **getValue of interface InstanceElement**(p. 222) use the stored values. So an instance of AoLocalColumn with a sequence_representation of implicit_linear will return exactly two values. These two values are also accessible with the baseattribute generation_parameters. If the sequence_representation tells the client the values are stored in an external component (external_component) the server will load the values from the external file and returns the values to the client when the client asks for them. If the values are raw values, e.g. sequence_representation is raw_linear, these methods return the raw values if requested for the values and the generation parameters will be retruns of the request for the generation_parameters. Only access to all values is possible.
2. Using the interface **ValueMatrix**(p. 270). With the methods of the ValueMatrix the access to the values is always to the calculated values. The sequence_representation of the AoLocalColumn makes no difference for the access and interpretation of the values by the client. Access to pieces of the values is possible. The names of the columns are the names of the corresponding AoMeasurementQuantities and not the names of the AoLocalColumns.

1.9 Usage of the value flags.

The basemodel defined for the base element AoLocalColumn two base attributes, **global_flag** and **flags**, which are used for the flags of the values. If the **global_flag** is defined then the **flags** can be ignored and all values will have the flag like the **global_flag**. As soon as the **global_flag** is undefined, the **flags** defines the flag for each value. The **flags** is a sequence of shorts (DS_SHORT), the length of the sequence is identical with the number of values. The index in the sequence match with the index of the values.

The field **flag** of the TS_Value is used for the state of the attribute. If this field is 0, the value of the structure is undefined. If this field is 15, the value of the structure is defined. Using this field the **global_flag** can be set in a defined to undefined state. If the **global_flag** is in a defined state, the client doesn't need to load the **flags**.

When the client writes the values and both the **global_flag** and **flags** are both in an undefined state, the server will automatically set the **global_flag** in a defined state and the attribute flag (**flag** field of TS_Value) of the values will be the value of the **global_flag**.

```
/*
 * 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.
 */

// The following example shows a defined global_flag with the value
// (AO_VF_VALID|AO_VF_VISIBLE|AO_VF_DEFINED)
NameValueUnit globalF = new NameValueUnit();
globalF.valName = "global_flag";      /* Name of the attribute, in this
                                     * example the name of the base
                                     * attribute is used.
                                     */
```

```

globalF.unit = ""; // There is no unit
globalF.value = new TS_Value();
globalF.value.u = new TS_Union();
globalF.value.u.shortVal((short)(1|2|8)); // All values are modified.
globalF.value.flag = (1|2|4|8); // global_flag is defined.

// The following example shows an undefined global_flag
NameValueUnit globalF = new NameValueUnit();
globalF.valName = "global_flag"; /* Name of the attribute, in this
                                * example the name of the base
                                * attribute is used.
                                */

globalF.unit = ""; // There is no unit
globalF.value = new TS_Value();
globalF.value.u = new TS_Union();
globalF.value.u.shortVal((short)(15)); // This value has no influence.
globalF.value.flag = 0; // global_flag is undefined.

```

1.10 Logging of the server behaviour

The ODS API implementation allows switching of the debuglevel and log-file during runtime. The ODS API implementation changes the debuglevel or the log-file when the context variables are written. The context variable `DEBUGLEVEL` changes the debuglevel. The context variable `ODS_LOGFILE` close the current log-file and opens a new log file. The `DEBUGLEVEL` and `ODS_LOGFILE` are identical with the INI-file variable, see for more detail information in the technical reference sheet.

The following lines shows how to change the debuglevel and the log-file.

```

session.setContextString("DEBUGLEVEL", "10"); // Set the debuglevel to 10.
session.setContextString("ODS_LOGFILE", "c:\\mylog.txt"); // close the current log-file
// open log-file c:\\mylog.txt

```

1.11 Connect to the ODS API using HighQSoft implementation.

The JAVA implementation of HighQSoft allows the client to use the same code for the direct binding with JAVA to the implementation of the CORBA- binding to an ODS API Server such as the HighQSoft Avalon. The class `AoFactory` is the entry point of the ASAM ODS API. An object of the class `AoFactory` will be published at the CORBA-Naming Service by the ODS API Server. The client have to connect to the CORBA-Naming Service and query for the published objects and bind to one of these objects. The HighQSoft defined interface `com.highqsoft.ods.AoService` hides the connect, list and bind to the CORBA-Naming Service. With the HighQSoft defined class `com.highqsoft.ods.AoServiceFactory` you can get an object implement this interface. There is also an implementation of the interface `com.highqsoft.ods.AoService` for the direct binding with JAVA.

1.12 AoServiceFactory

Package:

`com.highqsoft.ods`



Description:

The ASAM-ODS Service Factory. The ASAM-ODS Service Factory allows to deploy different ASAM-ODS transport services. This is essential to create pluggable ASAM-ODS components - no matter if these components use CORBA, local native calls or other communication (transport) mechanisms.

For an example of the usage of this class look at **AoFactory**(p. 23).

1.12.1 Static method newService of AoServiceFactory**Purpose:**

Creates a new service via the requested pluggable component.

Return:

[AoService](#) aoService *The requested ASAM-ODS service of the pluggable component.*

Parameter:

[T_STRING](#) pluggableComponent (in) *The name of the requested pluggable component. Two classes of HighQSoft are com.highqsoft.ods.corba.AoService or com.highqsoft.ods.athos.AoService.*

[T_STRING](#) serviceOptions (in) *Service dependent options.*

Example 1 (Athos implementation): "INIFILE=ATHOS_INI"

Example 2 (Corba implementation): "NameServicePort=2809, NameServiceHost=localhost"

Errors:

[AO_BAD_PARAMETER](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)

1.13 AoService**Package:**

com.highqsoft.ods

Description:

The ASAM Service interface.

The ASAM Service is a proposal to all ASAM groups for a identical entrypoint to connect to an ASAM service.

Within ASAM-ODS this interface will hide the communication layer and is part of the client software.

For an example of the usage of this class look at **AoFactory**(p. 23).

1.13.1 listServices of interface AoService**Purpose:**

Returns a list of available services.

Return:

[NameSequence](#) services *The names of the available services..*

Parameter:

[T_STRING](#) servicePattern (in) *The name or the search pattern of the requested service. The pattern "*" is equivalent to NULL.*

Errors:

[AO_BAD_PARAMETER](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)

1.13.2 newFactory of interface AoService

Purpose:

Creates a new session factory of the requested service.

Return:

[AoFactory](#) aoFactory *The factory of the service sessions.*

Parameter:

[T_STRING](#) serviceName (in) *The name of the service for which a new session factory shall be instantiated.*

[T_STRING](#) factoryOptions (in) *A string that contains optional name-value pairs, e.g. DEBUGLEVEL=3. The options are implementation dependent. An implementation must have reasonable defaults for proper operation.*

Errors:

[AO_BAD_PARAMETER](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)

1.14 Transaction handling in ODS API.

Depending on the mode of the translator there are different ways of transaction handling.

1.14.1 Transaction handling with full database storage

When using a translator with full database access, the server can use the transaction handling of the database when for each session an own database connection is configured. The methods of the interface [ApplElemAccess](#) will insert and update direct to the database. The modification of the application structure or the enumerations will be written to the database at the commit transaction. The modification will be available for the current session and all following new created session, the session open parallel will not see the modifications of the application model. The modification of the security information will go direct to the database. Create of the instances at the object oriented methods [createInstance of interface ApplicationElement](#) will write the new instance at the commit transaction to the database, query will not return the instances until

commit transaction is done. Update existing instances will update direct the database also with object oriented methods, so query on the modified attributes is possible. Modification of the values of the columns with the interface [ValueMatrix](#) will update direct the database or external component files with the database transaction handling, the external component files are part of the transaction handling. The values of the localcolumn will be copied to a new file and the existing file will be deleted when not used anymore by another localcolumn. Modification of instances [AoSubmatrix](#) and [AoLocalcolumn](#) with the methods [setValue](#) or [setValueSeq](#) of the interface [InstanceElement](#) will goes directly to the database, the new instances are stored at the end of the transaction. A mixture of writing/creating instances with the methods of the interface [AppElemAccess](#) and the object oriented methods can cause to problems because the objects created with the object oriented methods are not directly stored in the database storage.

1.14.2 Transaction handling with RPC-API translators and database storages

At commit transaction the instances are transfered to the server, query will return the new and update instances after commit transaction. Any other transaction handling is not possible due to the missing transaction handling in the RPC-API. The modifications of the values in the [valuematrix](#) are written direct to the server and there is no transaction handling.

1.14.3 Transaction handling with file based translators (ATF)

The transaction handling of the file bases translators is done in Athos Runtime System. The query is also done by the Athos Runtime System so direct after creating or modifying the instances the information is available for all accessing the datastorage. The information is written to the file when the service is closed. This is a close of the session and delete (deconstructor) of the session and the factory.



Chapter 2

Explanation of the documentaion

2.1 Explataion of the method documetation

The documentation of the methods in the **ASAM ODS Interface** have all the same fields, some of the fields are explained below.

2.1.1 Explanation of the Return.

Return:

NameValue contextVariable *The requested context variable.*

- NameValue The type of the return variable.
- contextVariable The name of the retrun variable.
- The description of the return paramater in *italic*

2.1.2 Explanation of the Parameter.

Parameter:

Name varName (in) *The name of the requested context variable.*

- Name The type of the parameter.
- varName The name of the parameter.
- (in) describes the direction of the parameter, teh drection can be 'in', 'out' or 'in/out' for in and out
- The description of the paramater in *italic*

2.1.3 Explanation of the Tested by.

Tested by:

In this paragraph test methods are listed which uses this method during the test. Not all test methods are in the list and when the paragraph don't exist it does not means the method is not test at all.

2.1.4 Explanation of the examples.

In the examples of this chapter, the Corba and AoException are not caught. This is required in the client programming but to leave the example smaller we decided to leave out the catch of the exception.

The example of the method getContext of the interface AoSession should be:

```
import org.asam.ods.AoSession;
import org.asam.ods.NameValue;
import org.asam.ods.NameValueIterator;

// Create a new session with aoFactory.
AoSession session=null;
try {
    session = aoFactory.newSession("");
    try {
        // 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 for 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()
                                    + "\"");
            }
        }
    } catch (AoException aoEx) {
    }
    // Close the session.
    session.close();
} catch (AoException aoEx) {
}
}
```


Chapter 3

Connect to an ASAM ODS server using the implementation of HighQSoft

An ASAM ODS Server publishes his factory of type **AoFactory**(p. 23) at an CORBA-Naming Service. At most CORBA implemetations there is example how to connect to the CORBA-Naming Service, this is also explained from the OMG. For the client development there is an implementation of the Highqsoft to connect to an ASAM ODS server using the CORBA-Naming Server.

3.1 Connect to the ODS API using HighQSoft implementation.

The JAVA implementation of HighQSoft allows the client to use the same code for the direct binding with JAVA to the implemetation of the CORBA- binding to an ODS API Server such as the HighQSoft Avalon. The class **AoFactory** is the entry point of the ASAM ODS API. An object of the class **AoFactory** will be published at the CORBA-Naming Service by the ODS API Server. The client have to connect to the CORBA-Naming Service and query for the published objects and bind to one of these objects. The HighQSoft defined interface `com.highqsoft.ods.AoService` hides the connect, list and bind to the CORBA-Naming Service. With the HighQSoft defined class `com.highqsoft.ods.AoServiceFactory` you can get an object implement this interface. There is also an implementation of the interface `com.highqsoft.ods.AoService` for the direct binding with JAVA.

3.1.1 Connect to an ATF-File

The implementation allows to connect to an ATF file. It makes no difference to open an ATF/CLA (classic) or ATF/XML file, the way to do is the same only another translator must be available in the service of the INI-file.

The following code snippet shows how to connect an ATF-file with java and the HighQSoft java language binding.

```
// Athos plugable modul
```

```
String plumAthos = "com.highqsoft.ods.athos.AoService";

// The option with the INIFILE
String option = "INIFILE=myAthos_ini_File";
// Get the service from the package 'plumAthos' with the services form the INI-file

AoService aoService = AoServiceFactory.newService(plumAthos, option);
AoFactory aoFac = aoService.newFactory(atfXmlService, null);

// Get the desired ASAM ODS session factory (without factory options),
// in the example "XATF/TestModel/Out".
AoFactory aoXmlFac = aoService.newFactory("XATF/TestModel/Out", null);

// Connect with a filename for reading
AoSession aoXmlSess = aoXmlFac.newSession("FILENAME=TestModelOut.xml, OPENMODE=r");
```

The example service in the INI-file is given below, important in the service is the line **REOPEN_VARIABLE = FILENAME** which force the implementation to reopen each time newSession is called the ATF file is reopened.

```
[SERVICE "XATF/TestModel/Out"]
DRIVER      = xatf
DIRECTORY   = "file:///$(ATHOS_ROOT)/bin/$(OSTYPE)/"
FILENAME    = "any.xml"
MODEL_FILENAME = "TestModel.xml"
NOSECURITYACTIVE = YES
IGNORE_SECURITY = YES
SEARCH_FOR_BASE_REF = NO
WRITEMODE=CHANNEL
REOPEN_VARIABLE = FILENAME
XATF_LOGFILE = "xatf_out.log"
ODS_LOGFILE = "xatf_ods_out.log"
ODSVERSION = "5.0 readonly"
LOAD_NEXT_ID = NO
```

When the service must be open to write in an ATF-File use at the method `newSession` the argument `OPENMODE=w`. With the above example INI-file the application model and the instances are read from the file given with the entry `MODEL_FILENAME` and the instances will be written to the file given with the `FILENAME` as argument of the method `newSession`. When no application model file is given the client must write the complete application model and all instances to the ATF file. The line **LOAD_NEXT_ID = NO** force the implementation to create Id's for the instances, so the client have to write also the Id's of the instances.

The session, factory and service must be closed and deleted properly otherwise the implementation will not write the ATF File. Languages with an automatic garbage collector like java will finalize (delete) the objects when garbage collectors runs. The implementation writes only the ATF-file when all the objects are deleted.

```
// Close the session
aoXmlSess.close();
aoXmlSess = null;
aoXmlFac = null;
aoService = null;
// Call the garbage collector to free the session and write the ATF file..
try {
    System.gc();
} catch (Throwable t) {
}
```



3.2 AoServiceFactory

Package:

com.highqsoft.ods

Description:

The ASAM-ODS Service Factory. The ASAM-ODS Service Factory allows to deploy different ASAM-ODS transport services. This is essential to create pluggable ASAM-ODS components - no matter if these components use CORBA, local native calls or other communication (transport) mechanisms.

For an example of the usage of this class look at **AoFactory**(p. 23).

3.2.1 Static method newService of AoServiceFactory

Purpose:

Creates a new service via the requested pluggable component.

Return:

[AoService](#) aoService *The requested ASAM-ODS service of the pluggable component.*

Parameter:

[T_STRING](#) plugableComponent (in) *The name of the requested pluggable component. Two classes of HighQSoft are com.highqsoft.ods.corba.AoService or com.highqsoft.ods.athos.AoService.*

[T_STRING](#) serviceOptions (in) *Service dependent options.*

Example 1 (Athos implementation): "INIFILE=ATHOS_INI"

Example 2 (Corba implementation): "NameServicePort=2809, NameServiceHost=localhost"

Errors:

[AO_BAD_PARAMETER](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)

3.3 AoService

Package:

com.highqsoft.ods

Description:

The ASAM Service interface.

The ASAM Service is the implementation of HighQSoft to connect to an object of **AoFactory**(p. 23) This interface hides the communication layer and is part of the client software.

For an example of the usage of this class look at **AoFactory**(p. 23).

There are some special features of the implementation to connect to

3.3.1 listServices of interface AoService

Purpose:

Returns a list of available services.

Return:

[NameSequence](#) services *The names of the available services..*

Parameter:

[T_STRING](#) servicePattern (in) *The name or the search pattern of the requested service. The pattern "*" is equivalent to NULL.*

Errors:

[AO_BAD_PARAMETER](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)

3.3.2 newFactory of interface AoService

Purpose:

Creates a new session factory of the requested service.

Return:

[AoFactory](#) aoFactory *The factory of the service sessions.*

Parameter:

[T_STRING](#) serviceName (in) *The name of the service for which a new session factory shall be instantiated.*

[T_STRING](#) factoryOptions (in) *A string that contains optional name-value pairs, e.g. DEBUGLEVEL=3. The options are implementation dependent. An implementation must have reasonable defaults for proper operation.*

Errors:

[AO_BAD_PARAMETER](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)

Chapter 4

ASAM ODS Interface

4.1 AoFactory

Package:

org.asam.ods

Description:

The ASAM factory interface.

The factory is the object published at the NameService. The Avalon publishes an object with the name of the service in the ATHOS_INI file. The object is published twice, in the root context with the kind "ASAM-ODS" and in the context "org/asam/ods" without any kind. If the name of the service in the ATHOS_INI file contains '/' they will be used as child contexts of the "org/asam/ods" context.

Note:

The description, type and interfaceVersion are not loaded from the instance of AoEnvironment as defined in ASAM ODS but are loaded from the INI-File.

The following example shows how the factory can be loaded using the Athos tools.

```
/* @(#) $Id: ExamplesConnect.java,v 1.9 2008/01/10 09:22:36 karst Exp $
*****
* COPYRIGHT I, 1996-2008
* Hans-Joachim Bothe, Andreas Hofmann, Karst Schaap, Michael Ziller
*****
* HighqSoft GmbH
* Schlossborner Weg 6b, D-61479 Glashuetten/Taunus, Germany
* Phone: +49 6174 62915, Fax: +49 6174 62935, Internet: www.highqsoft.com
*****
*
* All Rights Reserved.
*
* This software is the confidential and proprietary information of the
* authors. It may be freely copied and distributed with the following
* stipulations:
*
*   o No fee except to recover costs of media and delivery may
*     be charged for the use or possession of this software.
*   o Sources to this utility must be made available in machine-
*     readable form along with the executable form.
```

```

*      o No portion of this program may be used in any program sold      *
*      for a fee or used for production purposes.                        *
*      o This copyright notice must not be removed.                      *
*                                                                           *
* All brand names and product names used in this software are trademarks, *
* registered trademarks, or trade names of their respective holders.      *
* The authors of this software are not associated with any product or      *
* vendor mentioned in the code or documentation.                         *
*                                                                           *
*****
*/

import org.asam.ods.AoException;
import org.asam.ods.AoFactory;
import org.asam.ods.AoSession;

import com.highqsoft.ods.AoService;
import com.highqsoft.ods.AoServiceFactory;

/**
 *
 * <p>
 * <TABLE BORDER class="PropertyTable">
 * <TR><TH> Property Keyword </TH><TH> Datatype </TH><TH> Default </TH><TH> Description </TH></TR>
 * <TR><TD> ServiceName </TD>
 * <TD> String </TD>
 * <TD> "ZEUS/DB.ASAM-ODS". </TD>
 * <TD> The service to be used tool.<br>
 * <em>This value is an application property.</em> </TD></TR>
 * <TR><TD> WorkingDirectory </TD>
 * <TD> string </TD>
 * <TD> "." </TD>
 * <TD> The working directory to check the files. </TD></TR>
 * <TR><TD> UserAuthentication </TD>
 * <TD> string </TD>
 * <TD> "" </TD>
 * <TD> The user authentication. </TD></TR>
 * <TR><TD> PluggableComponent </TD>
 * <TD> string </TD>
 * <TD> "CORBA" </TD>
 * <TD> The working directory to check the files. </TD></TR>
 * <TR><TD> NoVerbose </TD>
 * <TD> string </TD>
 * <TD> "no" </TD>
 * <TD> Output of the connect information. </TD></TR>
 * </TABLE>
 *
 * @author Karst Schaap
 * @version $Revision: 1.9 $
 * @since $Date: 2008/01/10 09:22:36 $
 */

/**
 * Modification History:
 *
 * $Log: ExamplesConnect.java,v $
 * Revision 1.9 2008/01/10 09:22:36 karst
 * Change copyright to current.
 *
 * Revision 1.8 2005/09/22 11:59:38 andy
 * dd a main method to use it as a standalone testing tool.
 *
 * Revision 1.7 2005/04/26 15:13:13 karst
 * Organize the import list

```



```

*
* Revision 1.6 2005/01/07 15:55:30 karst
* Move the org.asam.ods.AoService to the directory com.highqsoft.ods.
* The org.asam.ods.AoService is a remaining of the ASAM ODS 4.0.
*
* Revision 1.5 2005/01/04 08:10:20 karst
* Change copyright to current.
*
* Revision 1.4 2004/09/20 08:00:46 karst
* Little correction, after remove of AoService.
*
* Revision 1.3 2004/09/20 07:45:40 karst
* Remove the import of the AoService (ODS 4.0).
*
* Revision 1.2 2004/05/21 08:17:12 karst
* Added the NoVerbose.
*
* Revision 1.1 2003/09/03 08:05:52 andy
* Move the source codes to the java subdirectory.
*
* Revision 1.1 2003/08/08 07:09:58 karst
* Initial version 3.0.
*
*
*/

public class ExamplesConnect {

    public static String getArgumentValue(String argName, String defVal) {

        String argValue = System.getProperty(argName, defVal);

        return(argValue);
    }

    public static AoSession connectService(String[] args) throws AoException {

        // Athos plugable modul
        String plumAthos = "com.highqsoft.ods.athos.AoService";
        // Corba plugable modul
        String plumCorba = "com.highqsoft.ods.corba.AoService";
        // Selected plugable modul
        String plum;
        // Convert the arguments to options and write them to the system properties.
        String options = Convert.argsToOptions(args);

        boolean verbose = true;
        String noVerbose = System.getProperty("NoVerbose", "no");
        if (noVerbose.compareToIgnoreCase("yes") == 0) {
            verbose = false;
        }

        if (verbose) {
            // Say in which example we are.
            System.out.println("\nExamplesConnect.connectService");
            System.out.println("-----");
        }

        // Select the plugable modul.
        String plugCom = System.getProperty("PlugableComponent", "CORBA");
        if (plugCom.compareToIgnoreCase("athos") == 0) {
            plum = plumAthos;
            if (verbose) {
                System.out.println("Connect to ATHOS: using: " + plum);
            }
        }
    }
}

```

```

    }
} else {
    plum = plumCorba;
    if (verbose) {
        System.out.println("Connect to CORBA: using: " + plum);
    }
}

AoFactory aoFactory = null;          // The ASAM ODS factory.
AoSession aoSession = null;         // The ASAM ODS session.

// Create a new service via the HighQSoft Athos or Corba client.
AoService aoService = AoServiceFactory.newService(plum,options);

String serviceName = System.getProperty("ServiceName", "ZEUS/DB.ASAM-ODS");
if (verbose) {
    System.out.println("Try to connect to service: " + serviceName);
}

// Get the desired ASAM ODS session factory (without factory options).
aoFactory = aoService.newFactory(serviceName, null);

// Establish a session to the service (session options).
String userLogin = System.getProperty("UserAuthentication", "");

if (verbose) {
    System.out.println("Try to connect to session with: " + userLogin);
}
aoSession = aoFactory.newSession(userLogin);

return aoSession;
}

public static void main (String args[]) {
    try {
        AoSession aoSession = connectService (args);
        if (aoSession != null) {
            org.asam.ods.ApplicationStructure asObj = aoSession.getApplicationStructure();
        } else {
            System.err.println ("Got no session from server.\nSee server site log file for more information.");
        }
    } catch (Throwable t) {
        t.printStackTrace();
    }
}
}

```

An example how to use the AoFactory interface is given below.

```

/* @(#) $Id: UsingAoFactory.java,v 1.7 2008/01/10 09:22:36 karst Exp $
*****
* COPYRIGHT I, 1996-2008
* Hans-Joachim Bothe, Andreas Hofmann, Karst Schaap, Michael Ziller
*****
* HighQSoft GmbH
* Schlossborner Weg 6b, D-61479 Glashuetten/Taunus, Germany
* Phone: +49 6174 62915, Fax: +49 6174 62935, Internet: www.highqsoft.com
*****
*
* All Rights Reserved.
*
* This software is the confidential and proprietary information of the
* authors. It may be freely copied and distributed with the following
* stipulations:
*
*   o No fee except to recover costs of media and delivery may
*     be charged for the use or possession of this software.
*

```




```

*      o Sources to this utility must be made available in machine-      *
*      readable form along with the executable form.                    *
*      o No portion of this program may be used in any program sold      *
*      for a fee or used for production purposes.                        *
*      o This copyright notice must not be removed.                      *
*                                                                           *
* All brand names and product names used in this software are trademarks, *
* registered trademarks, or trade names of their respective holders.      *
* The authors of this software are not associated with any product or      *
* vendor mentioned in the code or documentation.                          *
*                                                                           *
*****
*/
/**
 * ASAM ODS Example: Using AoFactory.
 *
 * @author      Hans-Joachim Bothe
 * @version     $Revision: 1.7 $
 * @since      $Date: 2008/01/10 09:22:36 $
 *
 * This example shows how to use the ASAM ODS
 * factory class AoFactory.
 */

import org.asam.ods.AoException;
import org.asam.ods.AoFactory;
import org.asam.ods.AoSession;

import com.highqsoft.ods.AoService;
import com.highqsoft.ods.AoServiceFactory;

public class UsingAoFactory {

    // Java application entry.
    public static void main(String[] args) {

        // Say in which example we are.
        System.out.println("\nExample UsingAoFactory...");
        System.out.println("-----");

        // Timestamp for performance check.
        long mSeconds = System.currentTimeMillis();

        // Create the ASAM ODS service via an ASAM ODS ServiceFactory.
        // The class AoServiceFactory is not yet part of the ASAM ODS
        // standard and therefore still proprietary. The first
        // parameter of the method "newService" is a plugable ASAM ODS
        // implementation. Any name of a class that implements the
        // ASAM ODS interface org.asam.ods.AoService may be passed
        // as first parameter. Make sure the specified class file can
        // be found in the Java classpath. The second parameter is
        // an implementation dependent string which allows to pass
        // proprietary service initialization information to the
        // service factory.
        //
        // Corba example:
        //     AoService aoService = AoServiceFactory.newService(
        //         "com.highqsoft.ods.corba.AoService",
        //         "CorbaImplementation=JacORB, jacorb.verbosity=3");
        //
        // Athos example:
        //     AoService aoService = AoServiceFactory.newService(
        //         "com.highqsoft.ods.athos.AoService",
        //         "INIFILE=ATHOS_INI");
        //
        try {

```

```

// Create a new service via the HighQSoft Athos or Corba client.
AoService aoService = AoServiceFactory.newService(
    "com.highqsoft.ods.athos.AoService",
    Convert.argsToOptions(args));

// Print a list of available services.
String names[] = aoService.listServices("*");
System.out.println("Available service:");
for (int i=0; i<names.length; i++) {
    System.out.println("    " + names[i] + ":");
}

// Get the desired ASAM ODS session factory (without factory options).
AoFactory aoFactory = aoService.newFactory("AOP3Server", null);

// ASAM ODS session factory found ?
if (aoFactory != null) {

    // The results of the factory interface.
    System.out.println("\nASAM ODS Factory results:");

    // Get the factory name.
    try {
        String name = aoFactory.getName();
        System.out.println("    Factory name      : " + name);
    } catch (AoException aoException) {
        System.err.println("\nUsingAoFactory, getName failed, " + aoException.toString() + ":" +
            "\n    " + aoException.reason);
    }

    // Get the factory type.
    try {
        String type = aoFactory.getType();
        System.out.println("    Factory type      : " + type);
    } catch (AoException aoException) {
        System.err.println("\nUsingAoFactory, getType failed, " + aoException.toString() + ":" +
            "\n    " + aoException.reason);
    }

    // Get the factory description.
    try {
        String description = aoFactory.getDescription();
        System.out.println("    Factory description: " + description);
    } catch (AoException aoException) {
        System.err.println("\nUsingAoFactory, getDescription failed, " + aoException.toString() + ":" +
            "\n    " + aoException.reason);
    }

    // Get the interface version.
    try {
        String interfaceVersion = aoFactory.getInterfaceVersion();
        System.out.println("    Interface version : " + interfaceVersion);
    } catch (AoException aoException) {
        System.err.println("\nUsingAoFactory, getInterfaceVersion failed, " + aoException.toString() + ":" +
            "\n    " + aoException.reason);
    }

    // Get a session.
    try {
        AoSession aoSession = aoFactory.newSession("");
        System.out.println("    Session          : " + aoSession);
        if (aoSession != null) aoSession.close();
    } catch (AoException aoException) {
        System.err.println("\nUsingAoFactory, newSession failed, " + aoException.toString() + ":" +
            "\n    " + aoException.reason);
    }
}
}

```

```

    } catch (AoException aoException) {
        System.err.println("\nUsingAoFactory, " + aoException.toString() + ":" +
            "\n    " + aoException.reason);
        if (aoException.reason.startsWith("org.omg.CosNaming.NamingContextPackage.NotFound")) {
            System.err.println("    Specified service not found. Check name in first parameter to:");
            System.err.println("        aoService.newFactory(\"<check_this_name>\", ...");
        }
    }

    // Print elapsed time.
    System.out.println ("\nElapsed Time (in ms): " + (System.currentTimeMillis()-mSeconds));

    // Exit the application.
    System.exit(0);
}
}

```

4.1.1 getDescription of interface AoFactory

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.

Note:

The description is loaded from the INI-File, using the variable DESCRIPTION of the service.

Return:

[T_STRING](#) description *The description of the ASAM ODS factory.*

Parameter:

None.

Java Calling Sequence:

```
T_STRING description = aoFactory.getDescription();
```

Java Example:

```
String description;
description = aoFactory.getDescription();
...
```

Errors:

[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)

Tested by:

```
test.highqsoft.avalon.AoFactoryTest.testDescription()
```

4.1.2 getInterfaceVersion of interface AoFactory

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-5.1'.

Note:

The interface version is loaded from the INI-File, using the variable ODSVERSION of the service.

Return:

[T_STRING](#) interfaceVersion *The interface version of the ASAM ODS factory.*

Parameter:

None.

Java Calling Sequence:

```
T_STRING interfaceVersion = aoFactory.getInterfaceVersion();
```

Java Example:

```
String interfaceVersion;  
interfaceVersion = aoFactory.getInterfaceVersion();
```

Errors:

[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)

Tested by:

```
test.highqsoft.avalon.AoFactoryTest.testInterfaceVersion()
```

4.1.3 getName of interface AoFactory

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.

Note:

The name of the service in the INI-File is returned.

Return:

[T_STRING](#) factoryName *The name of the ASAM ODS factory.*

Parameter:

None.

Java Calling Sequence:

```
T_STRING factoryName = aoFactory.getName();
```

Java Example:

```
String name;  
name = aoFactory.getName();
```

Errors:

[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)

Tested by:

```
test.highqsoft.avalon.AoFactoryTest.testName()
```

4.1.4 getType of interface AoFactory

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.

Note:

The type is loaded from the INI-File, using the variable TYPE of the service.

Return:

[T_STRING](#) factoryType *The type of the ASAM ODS factory.*

Parameter:

None.

Java Calling Sequence:

```
T_STRING factoryType = aoFactory.getType();
```

Java Example:

```
String type;  
type = aoFactory.getType();
```

Errors:

[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)

Tested by:

```
test.hiqsoft.avalon.AoFactoryTest.testType()
```

4.1.5 newSession of interface AoFactory

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.

Note:

The user/password is check, there is no check if the user have access rights of any element, this is done when the ApplicationStructure is loaded. At the 'auth' argument all context variables can be used in this string. There is a special configuration variable OPENMODE which defined the open mode of file translators. Use 'r' for read and 'w' for write.

Return:

[AoSession](#) aoSession *The new created ASAM ODS session.*

Parameter:

[T_STRING](#) auth (in) *A string that may contain authentication information. The following values are currently supported:*

USER

PASSWORD

CREATE_COSESSION_ALLOWED

*The values may be specified in any order and have to be separated by comma.
 The variable `CREATE_COSESSION_ALLOWED`
 Value (DT_STRING) `TRUE` or `FALSE`
 Default for `CREATE_COSESSION_ALLOWED` = `FALSE`.
 The variable `CREATE_COSESSION_ALLOWED` is a readonly variable in the session.
 Example:
 "USER=hans, PASSWORD=secret, CREATE_COSESSION_ALLOWED = TRUE"*

Java Calling Sequence:

```
AoSession aoSession = aoFactory.newSession(auth);
```

Java Example:

```
import org.asam.ods.AoSession;
AoSession session;
session = aoFactory.newSession("USER=hans, PASSWORD=secret");
```

Errors:

[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](#)

Tested by:

```
test.highqsoft.avalon.AoFactoryTest.setUp()
test.highqsoft.avalon.AoFactoryTest.testNewFactory()
test.highqsoft.avalon.AoSessionTest.setUp()
test.highqsoft.odsapi.read.AoFactoryTest.setUp()
test.highqsoft.odsapi.read.AoFactoryTest.testNewFactoryUser()
test.highqsoft.odsapi.CreateMeasurementTest.threadAsamPath.run()
test.highqsoft.odsapi.CreateMeasurementTest.threadConnect.run()
test.highqsoft.odsapi.XATFCopyTest.testCompareAllInstances()
test.highqsoft.odsapi.XATFCopyTest.testCompareInstances()
test.highqsoft.odsapi.XATFCopyTest.testCompareModel()
....
```

4.2 AoSession

Package:

org.asam.ods

Description:

The ASAM ODS session interface.

The objects created by the server stay in memory of the server until the session is closed. The destroy-methods (ASAM ODS 5.*) allows the server to remove the objects from memory. The server will remove all remaining objects from memory when the session is closed. The performance of the server is related to the number of objects in memory, the more objects in memory, the less the performance, so it is an advantage for the performance to create a new session for object intensive work and close the session again when the work is done.

Note:

The description, type and name are not loaded from the instance of AoEnvironment as defined in ASAM ODS but are loaded from the INI-File.

An example how to use the AoSession interface is given below.

```

/* @(#) $Id: UsingAoSession.java,v 1.8 2008/01/10 09:22:36 karst Exp $
*****
* COPYRIGHT I, 1996-2008
* Hans-Joachim Bothe, Andreas Hofmann, Karst Schaap, Michael Ziller
*****
* HighQSoft GmbH
* Schlossborner Weg 6b, D-61479 Glashuetten/Taunus, Germany
* Phone: +49 6174 62915, Fax: +49 6174 62935, Internet: www.highqsoft.com
*****
*
* All Rights Reserved.
*
* This software is the confidential and proprietary information of the
* authors. It may be freely copied and distributed with the following
* stipulations:
*
*   o No fee except to recover costs of media and delivery may
*     be charged for the use or possession of this software.
*   o Sources to this utility must be made available in machine-
*     readable form along with the executable form.
*   o No portion of this program may be used in any program sold
*     for a fee or used for production purposes.
*   o This copyright notice must not be removed.
*
* All brand names and product names used in this software are trademarks,
* registered trademarks, or trade names of their respective holders.
* The authors of this software are not associated with any product or
* vendor mentioned in the code or documentation.
*
*****
*/
/**
* ASAM ODS Example: Using AoSession.
*
* @author      Hans-Joachim Bothe
* @version     $Revision: 1.8 $
* @since      $Date: 2008/01/10 09:22:36 $
*
* This example shows how to use the ASAM ODS
* class AoSession.
*/

import org.asam.ods.AoException;
import org.asam.ods.AoSession;
import org.asam.ods.ApplicationElement;
import org.asam.ods.ApplicationStructure;
import org.asam.ods.ApplicationStructureValue;
import org.asam.ods.BaseStructure;
import org.asam.ods.NameValue;
import org.asam.ods.NameValueIterator;

/**
* Using AoSession.
* <br>
* All the methods of the asam ods AoSession-Interface are used in the example.
*
*/
public class UsingAoSession {

    // Java application entry.

```

```

public static void main(String[] args) {

    // Say in which example we are.
    System.out.println("\nExample UsingAoSession...");
    System.out.println("-----");

    // Timestamp for performance check.
    long mSeconds = System.currentTimeMillis();

    try {

        // Establish the session.
        AoSession aoSession = ExamplesConnect.connectService(args);

        try {

            // Get the application structure value.
            try {
                ApplicationStructureValue as = aoSession.getApplicationStructureValue();
                System.out.println("Application StructureValue: " + as);
            } catch (AoException aoException) {
                System.err.println("\nUsingAoSession, getApplicationStructureValue failed:" +
                                   "\n    " + aoException.toString() +
                                   "\n    " + aoException.reason);
            }

            // Get the application structure.
            try {
                ApplicationStructure as = aoSession.getApplicationStructure();
                System.out.println("Application Structure: " + as);
                ApplicationElement[] aes = as.getElements("*");
                if (aes != null) {
                    for (int i = 0; i < aes.length; i++) {
                        System.out.println(aes[i].getName());
                    }
                }
            } catch (AoException aoException) {
                System.err.println("\nUsingAoSession, getApplicationStructure failed:" +
                                   "\n    " + aoException.toString() +
                                   "\n    " + aoException.reason);
            }

            // Get the base structure.
            try {
                BaseStructure bs = aoSession.getBaseStructure();
                System.out.println("Base Structure:      " + bs);
            } catch (AoException aoException) {
                System.err.println("\nUsingAoSession, getBaseStructure failed:" +
                                   "\n    " + aoException.toString() +
                                   "\n    " + aoException.reason);
            }

            // Using context variables.

            // Get all known context variables.
            NameValueIterator vars = aoSession.getContext("*");
            int n = 0;
            if (vars != null) {
                n = vars.getCount();
            }
            System.out.println("\nNumber of context variables: " + n);

            // List all variable names and values.
            if (n > 0) {
                NameValue[] nvs = vars.nextN(n);
                for (int i=0; i<n; i++) {
                    System.out.println("    " + nvs[i].valName + " = " +

```



```

        nvs[i].value.u.stringVal());
    }
    System.out.println("");
}

// Set a new context variable.
String varName = "UsingAoSessionVariable";
aoSession.setContextString(varName, "The_test_value");

// Retrieve the new context variable value via iterator.
NameValueIterator varList = aoSession.getContext(varName);
if (varList != null) {
    System.out.println("Value of the test variable: " +
        varList.nextOne().value.u.stringVal());
}

// Retrieve the new context variable value by name.
NameValue var = aoSession.getContextByName(varName);
if (var != null) {
    System.out.println("Value of the test variable: " +
        var.value.u.stringVal());
}

// Remove the test context variable.
System.out.println("\nRemoving test variable...");
aoSession.removeContext(varName);

// Retrieve the new context variable value.
try {
    var = aoSession.getContextByName(varName);
    System.out.println("    " + var.valName + " = " +
        var.value.u.stringVal());
} catch (AoException aoex1) {
    if (aoex1.errCode.value() == org.asam.ods.ErrorCode._AO_NOT_FOUND) {
        System.out.println("The test variable does no longer exist.");
    } else {
        throw aoex1;
    }
}

// Using transactions. Randomly commit or abort.
System.out.println("\nTesting transaction methods:");
try {
    aoSession.startTransaction();
    System.out.println("\nExample transaction started.");
    if (Math.random() > 0.5) {
        aoSession.abortTransaction();
        System.out.println("Example transaction intentionally aborted.");
    } else {
        aoSession.commitTransaction();
        System.out.println("Example transaction committed.");
    }
} catch (AoException aoException) {
    System.err.println("\nUsingAoSession, transaction example failed:" +
        "\n    " + aoException.toString() +
        "\n    " + aoException.reason);
}

aoSession.close();
} catch (AoException aoException) {
    System.err.println("\nUsingAoSession, session close failed:" +
        "\n    " + aoException.toString() +
        "\n    " + aoException.reason);
}

} catch (AoException aoException) {
    System.err.println("\nUsingAoSession, " + aoException.toString() + ":" +

```

```

        "\n    " + aoException.reason);
    }

    // Print elapsed time.
    System.out.println ("\nElapsed Time (in ms): " + (System.currentTimeMillis()-mSeconds));

    // Exit the application.
    System.exit(0);
}
}

```

4.2.1 abortTransaction of interface AoSession

Purpose:

Abort (rollback) a transaction. The changes made in the transaction are lost.

Note:

See [Transaction handling in ODS API](#).(p. 14)

Return:

void

Parameter:

None.

Java Calling Sequence:

```
aoSession.abortTransaction();
```

Java Example:

```

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();
}

```

See also:

[startTransaction](#) of interface [AoSession](#)
[commitTransaction](#) of interface [AoSession](#)

Errors:

[AO_ACCESS_DENIED](#)



[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```

org.asam.ods.AoSessionPOA._invoke()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testAddAttribute()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemAltUnit()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemModAttr()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemModLength()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testDeleteAttribute()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testIeCreateRelatedInstances()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testModAttrLength()
....

```

4.2.2 close of interface AoSession**Purpose:**

Close session to an ASAM ODS server. Active transactions are committed.

Note:

Depending of the INI-File variable `SESSION_CLOSE_COMMIT` the active transaction is aborted or committed.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
aoSession.close();
```

Java Example:

```

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();
}

```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.AoFactoryTest.close()
test.highqsoft.avalon.AoFactoryTest.testNewFactory()
test.highqsoft.avalon.AoSessionTest.tearDown()
test.highqsoft.avalon.ExtQueryTest.runCatalogElements()
test.highqsoft.avalon.ExtQueryTest.runSimpleQueryExt()
test.highqsoft.avalon.ExtQueryTest.testCatalogElements()
test.highqsoft.avalon.ExtQueryTest.testCoSessCatalogElements()
test.highqsoft.avalon.ExtQueryTest.testCoSessSimpleQueryExt()
test.highqsoft.avalon.ExtQueryTest.testSimpleQueryExt()
test.highqsoft.odsapi.read.AoFactoryTest.close()
....

```

4.2.3 commitTransaction of interface AoSession**Purpose:**

Commit a transaction. The changes made in the transaction become permanent.

Note:

See [Transaction handling in ODS API](#).(p. 14)

Return:

void

Parameter:

None.

Java Calling Sequence:

```
aoSession.commitTransaction();
```

Java Example:

```

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();
}

```

See also:

[abortTransaction](#) of interface [AoSession](#)

[startTransaction](#) of interface [AoSession](#)



Errors:

[AO_ACCESS_DENIED](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testAddAttribute()
 test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
 test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
 test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateElement()
 test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemAltUnit()
 test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemModAttr()
 test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemModLength()
 test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemModObligatory()
 test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testDeleteAttribute()
 test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testIeCreateRelatedInstances()

4.2.4 createBlob of interface AoSession**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.

Return:

[Blob](#) blobObj *The reference of the blob object whgich is generated at the server.*

Parameter:

None.

Java Calling Sequence:

```
Blob blobObj = session.createBlob();
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.odsapi.UpdateInstanceElementTest.testBlobCompare()

4.2.5 createCoSession of interface AoSession**Purpose:**

Create a co session. The co session session is the same session as the original session was at login time.

The co session has no relation to the original session except that the same authentication is used.

A client application with components can give all components his own session to the ASAM ODS Server. So the component can close the session when it is ready, the component can start his own transaction without any conflict with other components.

Only when the variable `CREATE_COSESSION_ALLOWED=TRUE` is given in the 'auth' parameter of the method `newSession` at the interface `AoFactory` an new session will be created otherwise the exception `AO_ACCESS_DENIED` is thrown.

Return:

[AoSession](#) `coSession` *The co session.*

Parameter:

None.

Java Calling Sequence:

```
AoSession coSess = aoSess.createCoSession();
```

Since:

ASAM ODS 5.1

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_ACCESS_DENIED](#)

Tested by:

```
test.highqsoft.avalon.ApplicationStructureRelationTest.testApplicationStructureRelation()
test.highqsoft.avalon.ExtQueryTest.runCatalogElements()
test.highqsoft.avalon.ExtQueryTest.runSimpleQueryExt()
test.highqsoft.avalon.ExtQueryTest.testCatalogElements()
test.highqsoft.avalon.ExtQueryTest.testCoSessCatalogElements()
test.highqsoft.avalon.ExtQueryTest.testCoSessSimpleQueryExt()
test.highqsoft.avalon.ExtQueryTest.testSimpleQueryExt()
test.highqsoft.odsapi.UpdateMeasurementTest.testUpdateMeasurement()
```

4.2.6 createQueryEvaluator of interface AoSession

Purpose:

Create a `QueryEvaluator` object.

Note:

Not jet implemented, Query with query language not well defined in ASAM ODS.

Return:

[QueryEvaluator](#) `queryEvaluator` *The new created query evaluator object*

Parameter:

None.



Java Calling Sequence:

```
QueryEvaluator queryEvaluator = aoSession.createQueryEvaluator();
```

Errors:

```
AO_ACCESS_DENIED  
AO_BAD_PARAMETER  
AO_CONNECTION_LOST  
AO_IMPLEMENTATION_PROBLEM  
AO_NOT_IMPLEMENTED  
AO_NO_MEMORY  
AO_SESSION_NOT_ACTIVE
```

4.2.7 flush of interface AoSession**Purpose:**

Make the changes permanent.

Note:

Not yet implemented.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
aoSession.flush();
```

Java Example:

```
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();  
  
}
```

Errors:

[AO_ACCESS_DENIED](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

4.2.8 getApplElemAccess of interface AoSession**Purpose:**

Get the application element access object from the current session.

Return:

[ApplElemAccess](#) applElemAccess *The application element access object.*

Parameter:

None.

Java Calling Sequence:

```
ApplElemAccess applElemAccess = aoSession.getApplElemAccess();
```

Java Example:

```
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();
}
```

Errors:

[AO_ACCESS_DENIED](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.ApplicationStructureTest.testGetInstances()
test.highqsoft.avalon.ExtQueryTest.runCatalogElements()
test.highqsoft.avalon.ExtQueryTest.runSimpleQueryExt()
```




```

test.highqsoft.avalon.ExtQueryTest.testCatalogElements()
test.highqsoft.avalon.ExtQueryTest.testCoSessCatalogElements()
test.highqsoft.avalon.ExtQueryTest.testCoSessSimpleQueryExt()
test.highqsoft.avalon.ExtQueryTest.testSimpleQueryExt()
test.highqsoft.avalon.InstanceElementTest.testCheckRelLccMeq()
test.highqsoft.avalon.InstanceElementTest.testInstanceRelatedAgainstInverseRelated()
test.highqsoft.avalon.InstanceElementTest.testInstancesNamesQuery()
....

```

4.2.9 getApplicationStructure of interface AoSession

Purpose:

Get the application model from the current session.

Return:

[ApplicationStructure](#) applicationStructure *The application structure.*

Parameter:

None.

Java Calling Sequence:

```
ApplicationStructure applicationStructure = aoSession.getApplicationStructure();
```

Java Example:

```

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 Structure.
    ApplicationStructure as = session.getApplicationStructure();

    ...

    // Close the session.
    session.close();
}

```

Errors:

```

AO_ACCESS_DENIED
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

```

Tested by:

```

test.highqsoft.avalon.ApplicationRelationTest.testApplicationInverseRelationRange()
test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationBase()
test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationNames()

```

```

test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationStructure()
test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationType()
test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadApplicationElement.run()
test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadApplicationStructure-
Base.run()
test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadApplicationStructure-
Elements.run()
test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadTestEnumeration-
Definition.run()
test.highqsoft.avalon.ApplicationStructureRelationTest.ThreadGetRelations.run()
....

```

4.2.10 getApplicationStructureValue of interface AoSession

Purpose:

Get the application model as values from the current session.

Return:

[ApplicationStructureValue](#) applicationStructureValue *The application structure as value.*

Parameter:

None.

Java Calling Sequence:

```
ApplicationStructureValue applicationStructureValue = aoSession.getApplicationStructureValue();
```

Errors:

[AO_ACCESS_DENIED](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationStructure()
test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadTestEnumeration-
Definition.run()
test.highqsoft.avalon.ApplicationStructureRelationTest.ThreadGetRelations.run()
test.highqsoft.avalon.ApplicationStructureTest.testApplicationStructureValue()
test.highqsoft.avalon.ApplicationStructureTest.testEnumerationDefinition()
test.highqsoft.odsapi.read.ExtendedQueryTest.testTreeQueryExt()
test.highqsoft.odsapi.AAInsertInstanceTest.XXtestInsertInstances()
test.highqsoft.odsapi.CreateInstanceElementTest.testInsertInstances()
test.highqsoft.odsapi.UpdateMeasurementTest.testGenParExtCompMeasurement()
test.highqsoft.odsapi.UpdateMeasurementTest.testGenParMeasurement()
....

```

4.2.11 getBaseStructure of interface AoSession

Purpose:

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.



Return:

[BaseStructure](#) baseStructure *The base structure.*

Parameter:

None.

Java Calling Sequence:

```
BaseStructure baseStructure = aoSession.getBaseStructure();
```

Java Example:

```
"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 Structure.
    BaseStructure bs = session.getBaseStructure();

    ...

    // Close the session.
    session.close();

}"
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.BaseStructureTest.testBaseElement()
test.highqsoft.avalon.BaseStructureTest.testBaseElementTopLevel()
test.highqsoft.avalon.BaseStructureTest.testBaseRelations()
test.highqsoft.avalon.BaseStructureTest.testBaseStructureElements()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCheckAttribute()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testDeleteAttribute()
test.highqsoft.odsapi.ApplModelCreateTest.testCreateElements()
....
```

4.2.12 getContext of interface AoSession**Purpose:**

Get context variables from the session. A pattern string can be specified to select groups of variables.

Return:

[NameValueIterator](#) nvIterator *A list of context variables.*

Parameter:

[Pattern](#) varPattern (in) *The name or the search pattern for the context variable(s).*

Java Calling Sequence:

```
NameValueIterator nvIterator = aoSession.getContext(varPattern);
```

Java Example:

```
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 for 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();
}
```

See also:

[getContextByName](#) of interface [AoSession](#)
[listContext](#) of interface [AoSession](#)
[removeContext](#) of interface [AoSession](#)
[setContext](#) of interface [AoSession](#)
[setContextString](#) of interface [AoSession](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_FOUND](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.AoSessionTest.testAoSessionContextRead()
test.highqsoft.avalon.IteratorTest.testNameValueIteratorReset()
```

4.2.13 getContextByName of interface AoSession**Purpose:**

Get a context variable by its name from the session.



Return:

[NameValue](#) contextVariable *The requested context variable.*

Parameter:

[Name](#) varName (in) *The name of the requested context variable.*

Java Calling Sequence:

```
NameValue contextVariable = aoSession.getContextByName(varName);
```

Java Example:

```
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();
}
```

See also:

[getContext](#) of interface [AoSession](#)
[listContext](#) of interface [AoSession](#)
[removeContext](#) of interface [AoSession](#)
[setContext](#) of interface [AoSession](#)
[setContextString](#) of interface [AoSession](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_FOUND](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.AoFactoryTest.testDescription()
test.highqsoft.avalon.AoFactoryTest.testInterfaceVersion()
test.highqsoft.avalon.AoFactoryTest.testType()
test.highqsoft.avalon.AoSessionTest.testAoSessionContextFilter()
test.highqsoft.avalon.AoSessionTest.testAoSessionContextRead()
test.highqsoft.avalon.AoSessionTest.testAoSessionContextReadNrObjects()
test.highqsoft.avalon.AoSessionTest.testAoSessionContextWrite()
test.highqsoft.avalon.ValueMatrixTest.testCheckUnit()
test.highqsoft.avalon.ValueMatrixTest.testUnitConversion()
test.highqsoft.odsapi.AthosTestCase.getNoObjectsToDelete()
....
```

4.2.14 getDescription of interface AoSession**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.

Note:

The description is loaded from the INI-File, using the variable DESCRIPTION of the service.

Return:

[T_STRING](#) description *The description of the ASAM ODS session.*

Parameter:

None.

Java Calling Sequence:

```
T_STRING description = aoSession.getDescription();
```

Java Example:

```
String description;
description = aoSession.getDescription();
...
```

Errors:

[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_CONNECTION_LOST](#)

Tested by:

```
test.highqsoft.avalon.AoFactoryTest.testDescription()
test.highqsoft.avalon.AoFactoryTest.testNewFactory()
test.highqsoft.odsapi.read.AoFactoryTest.testNewFactoryUser()
```

4.2.15 getLockMode of interface AoSession

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.

Return:

[T_SHORT](#) lockMode *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.*

Parameter:

None.

Java Calling Sequence:

```
LockMode lockMode = aoSession.getLockMode();
```

Errors:

[AO_ACCESS_DENIED](#)
[AO_BAD_PARAMETER](#)



AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

4.2.16 getName of interface AoSession

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 name from the base attribute "name" of the instance of AoEnvironment.

Return:

Name sessionName *The name of the ASAM ODS session.*

Parameter:

None.

Java Calling Sequence:

```
Name sessionName = aoSession.getName();
```

Java Example:

```
String name;  
name = aoSession.getName();
```

Errors:

AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
AO_CONNECTION_LOST

Tested by:

```
test.highqsoft.avalon.AoFactoryTest.testName()  
test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationStructure()
```

4.2.17 getType of interface AoSession

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.

Note:

The type is loaded from the INI-File, using the variable TYPE of the service.

Return:

T_STRING sessionType *The type of the ASAM ODS session.*

Parameter:

None.

Java Calling Sequence:

```
T_STRING sessionType = aoSession.getType();
```

Java Example:

```
String type;
type = aoSession.getType();
```

Errors:

[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_CONNECTION_LOST](#)

Tested by:

```
test.highqsoft.avalon.AoFactoryTest.testType()
```

4.2.18 getUser of interface AoSession**Purpose:**

Returns the instance element of the user logged in, this is the instance element from the application element derived from AoUser, with the name given in s given in the 'auth' parameter of the method newSession at the interface AoFactory (variable USER).

When the client settings of the user are stored in the ASAM ODS application model, this methods helps the client to get his settings.

Return:

[InstanceElement](#) ieUser *The instance element of the logged in user.*

Parameter:

None.

Java Calling Sequence:

```
InstanceElement ieUser = aoSession.getUser();
```

Since:

ASAM ODS 5.1

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)

Tested by:

```
test.highqsoft.odsapi.read.AoFactoryTest.testNewFactoryUser()
test.highqsoft.odsapi.TestConnectServer.testConnectServer()
```



4.2.19 listContext of interface AoSession

Purpose:

List the names of context variables from the session. A pattern string can be specified to select groups of variables.

Return:

[NameIterator](#) nameIterator *A list of context variable names.*

Parameter:

[Pattern](#) varPattern (in) *The name or the search pattern for the context variable(s).*

Java Calling Sequence:

```
NameIterator nameIterator = aoSession.listContext(varPattern);
```

See also:

[getContext](#) of interface [AoSession](#)
[getContextByName](#) of interface [AoSession](#)
[removeContext](#) of interface [AoSession](#)
[setContext](#) of interface [AoSession](#)
[setContextString](#) of interface [AoSession](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NOT_FOUND](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.AoSessionTest.testAoSessionContextRead()  
test.highqsoft.odsapi.XATFCopyTest.testCompareSecurityInformation()
```

4.2.20 removeContext of interface AoSession

Purpose:

Remove context variables from the session. A pattern string can be specified to remove groups of variables.

Return:

void

Parameter:

[Pattern](#) varPattern (in) *The name or the search pattern for the context variable(s) to be removed.*

Java Calling Sequence:

```
aoSession.removeContext(varPattern);
```

Java Example:

```

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();

}

```

See also:

[getContext](#) of interface [AoSession](#)
[getContextByName](#) of interface [AoSession](#)
[listContext](#) of interface [AoSession](#)
[setContext](#) of interface [AoSession](#)
[setContextString](#) of interface [AoSession](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NOT_FOUND](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.AoSessionTest.testAoSessionContextWrite()
test.highqsoft.odsapi.CreateMeasurementTest.testCreateLongMeasurement()

4.2.21 setContext of interface AoSession**Purpose:**

Set/modify a known context variable or add a new context variable to the session.

Return:

void

Parameter:

[NameValue](#) contextVariable (in) *The context variable.*



Java Calling Sequence:

```
aoSession.setContext(contextVariable);
```

Java Example:

```
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 = 15;
    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();
}
```

See also:

[getContext](#) of interface [AoSession](#)
[getContextByName](#) of interface [AoSession](#)
[listContext](#) of interface [AoSession](#)
[removeContext](#) of interface [AoSession](#)
[setContextString](#) of interface [AoSession](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.AoSessionTest.testAoSessionContextWrite()
```

4.2.22 setContextString of interface AoSession**Purpose:**

Set/modify a known context variable or add a new context variable to the session. This is a convenience method for the frequently used string variable type. It uses setContext internally.

Note:

Use this method to set the DEBUGLEVEL and the ODS_LOGFILE during runtime.

Return:

void

Parameter:

[Name](#) varName (in) *The name of the context variable.*

[T_STRING](#) value (in) *The value of the context variable.*

Java Calling Sequence:

```
aoSession.setContextString(varName,value);
```

Java Example:

```
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();

}
```

See also:

[getContext](#) of interface [AoSession](#)
[getContextByName](#) of interface [AoSession](#)
[listContext](#) of interface [AoSession](#)
[removeContext](#) of interface [AoSession](#)
[setContext](#) of interface [AoSession](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.AoSessionTest.testAoSessionContextWrite()
test.highqsoft.odsapi.CreateInstanceElementTest.testCreateMeaInstances()
test.highqsoft.odsapi.CreateInstanceElementTest.testUpdateMeaInstances()
test.highqsoft.odsapi.CreateMeasurementTest.testAddColumn()
test.highqsoft.odsapi.CreateMeasurementTest.testCreateLongMeasurement()
```



```
test.highqsoft.odsapi.CreateMeasurementTest.testCreateStringMeasurement()
test.highqsoft.odsapi.StopServer.testStopServer()
test.highqsoft.odsapi.CreateMeasurementTest.testManyMeasurement()
test.highqsoft.odsapi.DeleteInstanceElementTest.testDeleteInstances()
```

4.2.23 setCurrentInitialRights of interface AoSession

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.

Note:

This method doesn't work for nested server. The setCurrentInitialRights method or function will not be called by the driver.

Return:

void

Parameter:

[InitialRightSequence](#) irlEntries (in) *The current initial rights.*

[T_BOOLEAN](#) set (in) *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);
```

Errors:

[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](#)

Tested by:

```
test.highqsoft.odsapi.security.InstanceCheckTest.testCurrentRights()
```

4.2.24 setLockMode of interface AoSession

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.

Note:

Not yet implemented.

Return:

void

Parameter:

T_SHORT lockMode (in) *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);
```

Errors:

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

4.2.25 setPassword of interface AoSession**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 the 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.

Note:

This method doesn't work for nested server, except when the user is a superuser. The setPassword method / function isn't used for the nested server but an update of the user instance is performed.

Return:

void

Parameter:

T_STRING username (in) *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.*

T_STRING oldPassword (in) *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.*

T_STRING newPassword (in) *The new password of the user.*

Java Calling Sequence:

```
aoSession.setPassword(username,oldPassword,newPassword);
```



Errors:

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

Tested by:

test.hiqsoft.odsapi.security.InstanceCheckTest.testSetPassword()

4.2.26 startTransaction of interface AoSession

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.

Note:

See [Transaction handling in ODS API](#).(p. 14)

Return:

void

Parameter:

None.

Java Calling Sequence:

```
aoSession.startTransaction();
```

Java Example:

```
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();
}
```



```

* All Rights Reserved.
*
* This software is the confidential and proprietary information of the
* authors. It may be freely copied and distributed with the following
* stipulations:
*
*   o No fee except to recover costs of media and delivery may
*     be charged for the use or possession of this software.
*   o Sources to this utility must be made available in machine-
*     readable form along with the executable form.
*   o No portion of this program may be used in any program sold
*     for a fee or used for production purposes.
*   o This copyright notice must not be removed.
*
* All brand names and product names used in this software are trademarks,
* registered trademarks, or trade names of their respective holders.
* The authors of this software are not associated with any product or
* vendor mentioned in the code or documentation.
*
*****
*/

import java.util.Vector;

import org.asam.ods.AIDName;
import org.asam.ods.AIDNameUnitId;
import org.asam.ods.AIDNameValueUnitId;
import org.asam.ods.AoException;
import org.asam.ods.AoSession;
import org.asam.ods.ApplElemAccess;
import org.asam.ods.ApplicationAttribute;
import org.asam.ods.ApplicationElement;
import org.asam.ods.ApplicationRelation;
import org.asam.ods.ApplicationStructure;
import org.asam.ods.ElemId;
import org.asam.ods.ElemResultSet;
import org.asam.ods.NameValueSeqUnitId;
import org.asam.ods.QueryStructure;
import org.asam.ods.RelationRange;
import org.asam.ods.Relationship;
import org.asam.ods.SelOpcode;
import org.asam.ods.SelOperator;
import org.asam.ods.SelOrder;
import org.asam.ods.SelValue;
import org.asam.ods.TS_Union;
import org.asam.ods.TS_Value;
import org.asam.ods.T_LONGLONG;

import com.highqsoft.odsx.OdsxHelper;

/*
* <TABLE BORDER class="PropertyTable">
* <TR><TH> Property Keyword </TH><TH> Datatype </TH><TH> Default </TH><TH> Description </TH></TR>
* <TR><TD> ApplicationElementName </TD><TD> </TD>
* <TD> String </TD>
* <TD> "Messung". </TD>
* <TD> The name of the application element.<br>
* <em>This value is an application property.</em> </TD></TR>
* </TABLE>
*
*/

public class table_query {

    /* showElemResultSet
    * This method shows the result set of a query.
    *

```

```

    * param elemRes the result set o a query
    */
    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);
                System.out.println(OdsxHelper.ts_valueSeqToString(elemRes[i].attrValues[j].attrValues.value));
            }
        }
    }

    /* showTableResultSet
    *
    * Show the names of the element ans the names of the related instances.
    *
    * param elemRes the names and the ids of the references instances,
    *             the first attribute result set ist the name, the next
    *             attribute result set are the Ids.
    *             This method is complete client performance, no request to the server.
    * param elemResArray The vector with the element result sets of the references elements.
    *             The order of the results set is identical with the oder of the references
    *             attributes in the elemRes. All result set have two attributes, the first the
    *             attribute is the Id, the second attribute is the name.
    */
    public static void showTableResultSet (ElemResultSet elemRes, Vector elemResArray) {

        System.out.println("");

        // Get the names of the element.
        String names[] = elemRes.attrValues[0].attrValues.value.u.stringVal();
        int noInst = names.length;
        ElemResultSet relRes;

        // Get the list with the Ids
        int relIdArr[][] = new int[elemRes.attrValues.length-1][];
        for (int j = 0; j < relIdArr.length; j++) {
            relIdArr[j] = elemRes.attrValues[j+1].attrValues.value.u.longVal();
        }

        // i is the index of the rows
        for (int i = 0; i < noInst; i++) {
            // j is the index of the columns
            System.out.print(names[i]);
            for (int j = 0; j < relIdArr.length; j++) {
                // Get the result set from the vector
                relRes = (ElemResultSet) elemResArray.get(j);
                // Get the sequence with Ids and the sequences with names.
                // This are structure so client internal calls without a network roundtrip
                int relIds[] = relRes.attrValues[0].attrValues.value.u.longVal();
                String relNames[] = relRes.attrValues[1].attrValues.value.u.stringVal();

                // Get the Id of the related instance
                int relId = relIdArr[j][i];
                if (relId != 0) {
                    int k;
                    // Search for the related instance
                    for (k = 0; k < relIds.length; k++) {
                        if (relId == relIds[k]) {
                            System.out.print(", " + relNames[k]);
                            k = -3;
                            break;
                        }
                    }
                }
                if (k >= 0) {
                    System.out.print(", not found " + k);
                }
            }
        }
    }

```

```

        } else {
            // No related instance defined.
            System.out.print(", UNDEFINED");
        }
    }
    System.out.println("");
}
}

/* getRelAttributeName
 * This method returns the name of the relation attribute.
 * If the relation have the wrong relation range a null will be returned.
 * The relation range is important, because the attribute should resolve to one
 * instance element.
 *
 * param as the application structure
 *
 * param aeObj the source application element.
 *
 * param aeRel the related application element
 *
 * returns the name of the relation attribute
 */
public static String getRelAttributeName (ApplicationStructure as, ApplicationElement aeObj, ApplicationElement aeRel) {
    ApplicationRelation rel[] = as.getRelations(aeObj, aeRel);
    // Expect there is only one relation
    String relAttrName = null;
    RelationRange relRange = rel[0].getRelationRange();
    // Only n:1 relations.
    if (relRange.max == 1) {
        relAttrName = rel[0].getRelationName();
    }
    return (relAttrName);
}

/* getRelNames
 * Get the Ids and Names of the related instances. A Query is used to ask for the names, the Ids are the selection criteria.
 * The Ids are not reduced to a unique list of Ids, the server or database will do this.
 *
 * param aoSession, the current session. The application structure and applelemeaccess could be passed instead.
 *
 * param aeRel, the referenced application element
 *
 * param ids the list with the requested ids.
 */
public static ElemResultSet getRelNames(AoSession aoSession, ApplicationElement aeRel, NameValueSeqUnitId ids) throws Exception {
    ElemResultSet elemResRel[] = null;
    // Get the application structure, can be optimized
    ApplicationStructure as = aoSession.getApplicationStructure();

    // Get the appl elem access object, can be optimized
    ApplElemAccess aea = aoSession.getApplElemAccess();

    // Create the query structure for the method getInstances
    QueryStructure aoq;

    aoq = new QueryStructure();

    ApplicationAttribute aaObj;

    T_LONGLONG id = new T_LONGLONG();
    id.low = 0;
    id.high = 0;

    System.out.println("Load the names of the instances of " + aeRel.getName());
    // Report the id and name of the related instances.

```

```

aaObj = aeRel.getAttributeByBaseName("id");
aoq.anuSeq = new AIDNameUnitId [2];
aoq.anuSeq[0] = new AIDNameUnitId ();
aoq.anuSeq[0].attr = new AIDName();
aoq.anuSeq[0].attr.aid = aeRel.getId();
aoq.anuSeq[0].attr.aaName = aaObj.getName();
aoq.anuSeq[0].unitId = id;

aaObj = aeRel.getAttributeByBaseName("name");
aoq.anuSeq[1] = new AIDNameUnitId ();
aoq.anuSeq[1].attr = new AIDName();
aoq.anuSeq[1].attr.aid = aeRel.getId();
aoq.anuSeq[1].attr.aaName = aaObj.getName();
aoq.anuSeq[1].unitId = id;

// There is one condition
// The Id inset of the resultes in the attrResultSet

aaObj = aeRel.getAttributeByBaseName("id");
aoq.condSeq = new SelValue[1];
aoq.condSeq[0] = new SelValue();
aoq.condSeq[0].attr = new AIDNameValueUnitId();
aoq.condSeq[0].attr.attr = new AIDName();
aoq.condSeq[0].attr.attr.aid = aeRel.getId();
aoq.condSeq[0].attr.attr.aaName = aaObj.getName();
aoq.condSeq[0].attr.unitId = id;
aoq.condSeq[0].attr.values = new TS_Value();
aoq.condSeq[0].attr.values.flag = 15;
aoq.condSeq[0].attr.values.u = new TS_Union();
aoq.condSeq[0].attr.values.u.longVal(0); // Dummy

// The inset
aoq.condSeq[0].oper = SelOpcode.INSET;

aoq.condSeq[0].value = new TS_Value();
aoq.condSeq[0].value.flag = 15;
aoq.condSeq[0].value.u = new TS_Union();
aoq.condSeq[0].value.u.longSeq(ids.value.u.longVal()); // The value of TS_UnionSeq is identical with the Seq

// Initialize the not used fields

aoq.operSeq = new SelOperator[0];
aoq.orderBy = new SelOrder[0];
aoq.relInst = new ElemId();
aoq.relInst.aid = id;
aoq.relInst.iid = id;
aoq.relName = "";

elemResRel= aea.getInstances(aoq, 1000);
// showElemResultSet(elemResRel);

return (elemResRel[0]);
}

// Java application entry.
public static void main(String[] args) {

    // Say in which example we are.
    System.out.println("\nExample table_query...");
    System.out.println("-----");

    // Timestamp for performance check.
    long mSeconds = System.currentTimeMillis();

    try {

        // Establish the session.

```

```

AoSession aoSession = ExamplesConnect.connectService(args);

try {

    // Get the application structure.
    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[];

    // Query on the application element AoTest

    String aeName = ExamplesConnect.getArgumentValue("ApplicationElementName", "Messung");
    ApplicationElement aeObj = as.getElementByName(aeName);

    // Get all related elements.
    ApplicationElement ae[] = aeObj.getRelatedElementsByRelationship(Relationship.ALL_REL);

    int noRel = ae.length;

    // Create a list with names and relations.
    ApplicationElement aeRels[] = new ApplicationElement[noRel];
    String relAttrNames[] = new String[noRel];
    int relIndex = 0;
    for (int i = 0; i < noRel; i++) {
        // Get the name of the reference attribute.
        // Only with the correct cardinality will be returned.
        relAttrNames[relIndex] = getRelAttributeName(as, aeObj, ae[i]);
        if (relAttrNames[relIndex] != null) {
            aeRels[relIndex] = ae[i];
            relIndex++;
        }
    }
    noRel = relIndex;    // Set the number of used relations.

    T_LONGLONG id = new T_LONGLONG();
    id.low = 0;
    id.high = 0;

    // *****
    // Report the name attribute of the element
    aoq = new QueryStructure();

    // Report the name and for each related element the Id, or relation attribute
    aaObj = aeObj.getAttributeByBaseName("name");
    aoq.anuSeq = new AIDNameUnitId [1+noRel];
    aoq.anuSeq[0] = new AIDNameUnitId ();
    aoq.anuSeq[0].attr = new AIDName();
    aoq.anuSeq[0].attr.aid = aeObj.getId();
    aoq.anuSeq[0].attr.aaName = aaObj.getName();
    aoq.anuSeq[0].unitId = id;

    relIndex = 0;
    // Add the
    for (int i = 0; i < noRel; i++) {
        relIndex++;
        aoq.anuSeq[relIndex] = new AIDNameUnitId ();
        aoq.anuSeq[relIndex].attr = new AIDName();
    }
}

```

```

        // set Id of element
        aoq.anuSeq[relIndex].attr.aid = aeObj.getId();
        aoq.anuSeq[relIndex].attr.aaName = relAttrNames[i];
        aoq.anuSeq[relIndex].unitId = id;
    }

    // No Conditions
    aoq.condSeq = new SelValue[0];

    aoq.operSeq = new SelOperator[0];

    aoq.orderBy = new SelOrder[0];

    // Initialize the not used fields

    aoq.relInst = new ElemId();
    aoq.relInst.aid = id;
    aoq.relInst.iid = id;
    aoq.relName = "";

    elemRes = aea.getInstances(aoq, 1000);
    // showElemResultSet(elemRes);

    // Now we have the name and Id's of the related instances.
    // Load the ids and names of the related instances.
    ElemResultSet elemResRel;
    Vector elemResArray = new Vector();
    if ((elemRes != null) && (elemRes.length > 0)) {
        int noAttr = elemRes[0].attrValues.length;
        // The first attribute is the name all the other attributes are reference attributes, the order of the
        // attributes are identical with the request order.
        relIndex = 0;
        for (int i = 1; i < noAttr; i++) {
            elemResRel = getRelNames(aoSession, aeRels[relIndex], elemRes[0].attrValues[i].attrValues);
            relIndex++;
            elemResArray.add(elemResRel);
        }
    }

    // Print elapsed time.
    System.out.println("\nElapsed Time (in ms) for the query: " + (System.currentTimeMillis()-mSeconds));

    // Now we have the elemRes, with the instances of the application element and
    // the ids and names of the related instances in elemResArray.
    showTableResultSet(elemRes[0], elemResArray);

    aea = null;
    as = null;

} catch (AoException aoException) {
    System.err.println("\ntable_query, " +
        aoException.toString() + ":" +
        "\n    " + aoException.reason);
}

// Close the active session.
aoSession.close();

} catch (AoException aoException) {
    System.err.println("\ntable_query, " +
        aoException.toString() + ":" +
        "\n    " + aoException.reason);
}

System.out.println("\nTotal elapsed Time (in ms): " + (System.currentTimeMillis()-mSeconds));
// Exit the application.
System.exit(0);

```

```

    }
}

```

4.3.1 The return structure of the method getInstanceExt

The method **getInstanceExt** of interface **ApplElemAccess** (p. 81) returns a sequence of the structure **ResultSetExt** (see also **Definitions of the ResultSetExt structure.** (p. 310)). This sequence will always have the length of 1. This element in the structure has two fields

1. **firstElems** of type **ElemResultSetExtSequence** (see also [ElemResultSetExtSequence](#))
2. **restElems** of type **ElemResultSetExtSeqIterator** (see also **ElemResultSetExtSeqIterator** (p. 203))

firstElems

The **firstElems** is a sequence of **ElemResultSetExt** (see also **Definitions of the ElemResultSetExt structure.** (p. 304)), the length of the sequence corresponds with the number of different application elements requested in the **anuSeq** of the **QueryStructExt** (see also **asamods_structure_QueryStructExt**). The structure **ElemResultSetExt** has the fields **aid** and **values**. The **aid** is the Id of the application element, the **values**, a sequence of the structure **NameValueSeqUnitId** (see also **Definitions of the NameValueSeqUnitId structure.** (p. 306)) are the values of the attributes. For each reported attribute an entry is given in the sequence, the name of the attribute is given in the field **valName**. The Id of the unit is given in the field **unitId**, this is the Id of the instance of the application element derived from the base element **AoUnit**. The values are given in the field **value** with the type **TS_ValueSeq** (see also **Definitions of the TS_ValueSeq structure.** (p. 314)). Each value sequence has the same length, this length is maximum **how_many** the parameter of the method. If no values are found the length of the sequence is 0. The values with the same index in the different sequences of the entry in the **NameValueSeqUnitIdSequence** belongs to the same instance element.

restElems

The **restElems** is an iterator for the remaining sequence of **ElemResultSetExt**. This iterator allows to load the next N values from the **ResultSet**. The result of the element is the same way stored as at the field **firstElems**, the next N is the length of the sequence of the field **value** of the **NameValueSeqUnitId**. The method **nextOne** is wrong specified and should not be used, use instead the method **nextN** with **n=1**. The method **getCount** return the total number of found element, also the number already returned in **firstElems**. The method **reset** set the iterator the the beginning of the **ResultSet**, this is the beginning of the **firstElems**.

An example how to use the method **getInstanceExt** of the interface **ApplElemAccess** is given below.

```

/* @(#) $Id: table_query_ext.java,v 1.2 2008/01/10 09:22:37 karst Exp $
*****
* COPYRIGHT I, 1996-2008
* Hans-Joachim Bothe, Andreas Hofmann, Karst Schaap, Michael Ziller
*****
* HighQSoft GmbH
* Schlossborner Weg 6b, D-61479 Glashuetten/Taunus, Germany
* Phone: +49 6174 62915, Fax: +49 6174 62935, Internet: www.highqsoft.com
*****
*
* All Rights Reserved.
*

```

```

* This software is the confidential and proprietary information of the
* authors. It may be freely copied and distributed with the following
* stipulations:
*
*   o No fee except to recover costs of media and delivery may
*     be charged for the use or possession of this software.
*   o Sources to this utility must be made available in machine-
*     readable form along with the executable form.
*   o No portion of this program may be used in any program sold
*     for a fee or used for production purposes.
*   o This copyright notice must not be removed.
*
* All brand names and product names used in this software are trademarks,
* registered trademarks, or trade names of their respective holders.
* The authors of this software are not associated with any product or
* vendor mentioned in the code or documentation.
*
*****
*/

import org.asam.ods.AIDName;
import org.asam.ods.AIDNameUnitId;
import org.asam.ods.AggrFunc;
import org.asam.ods.AoException;
import org.asam.ods.AoSession;
import org.asam.ods.ApplElemAccess;
import org.asam.ods.ApplicationAttribute;
import org.asam.ods.ApplicationElement;
import org.asam.ods.ApplicationRelation;
import org.asam.ods.ApplicationStructure;
import org.asam.ods.JoinDef;
import org.asam.ods.JoinType;
import org.asam.ods.QueryStructureExt;
import org.asam.ods.RelationRange;
import org.asam.ods.ResultSetExt;
import org.asam.ods.SelAIDNameUnitId;
import org.asam.ods.SelItem;
import org.asam.ods.SelOpcode;
import org.asam.ods.SelOrder;
import org.asam.ods.SelValueExt;
import org.asam.ods.TS_Union;
import org.asam.ods.TS_Value;
import org.asam.ods.T_LONGLONG;

/*
 * <TABLE BORDER class="PropertyTable">
 * <TR><TH> Property Keyword </TH><TH> Datatype </TH><TH> Default </TH><TH> Description </TH></TR>
 * <TR><TD> ApplicationElementName </TD>
 * <TD> String </TD>
 * <TD> "Messung". </TD>
 * <TD> The name of the application element.<br>
 * <em>This value is an application property.</em> </TD></TR>
 * </TABLE>
 *
 */

public class table_query_ext {

    /**
     * Show the table of the extended query.
     *
     * @param elemRes the result set of the element.
     */
    public static void showTableResult(ResultSetExt elemRes) {
        int noRows;
        int noCols;
        if (elemRes.firstElems != null) {

```



```

        noCols=elemRes.firstElems.length;
        String vals[][] = new String[noCols][];
        short flags[][] = new short[noCols][];
        for (int colIndex = 0; colIndex < noCols; colIndex++) {
            vals[colIndex] = elemRes.firstElems[colIndex].values[0].value.u.stringVal();
            flags[colIndex] = elemRes.firstElems[colIndex].values[0].value.flag;
        }
        if (noCols > 0) {
            noRows = vals[0].length;
            System.out.println("Number of rows: " + noRows);
            for (int rowIndex = 0; rowIndex < noRows; rowIndex++) {
                for (int colIndex = 0; colIndex < noCols; colIndex++) {
                    if (flags[colIndex][rowIndex] == 15) {
                        System.out.print(vals[colIndex][rowIndex]+ " ");
                    } else {
                        System.out.print("UNDEFINED, ");
                    }
                }
                System.out.println();
            }
        }
    }
}

// Java application entry.
public static void main(String[] args) {

    // Say in which example we are.
    System.out.println("\nExample table_query_ext...");
    System.out.println("-----");

    // Timestamp for performance check.
    long mSeconds = System.currentTimeMillis();

    try {

        // Establish the session.
        AoSession aoSession = ExamplesConnect.connectService(args);

        try {

            // Get the application structure.
            ApplicationStructure as = aoSession.getApplicationStructure();

            // Get the appl elem access object
            ApplElemAccess aea = aoSession.getApplElemAccess();

            // Create the query structure for the method getInstances
            QueryStructureExt aoq;

            // Result of the request
            ResultSetExt elemRes[];

            // The application attribute.
            ApplicationAttribute aaObj;

            // Query on the application element AoTest

            String aeName = ExamplesConnect.getArgumentValue("ApplicationElementName", "Messung");
            ApplicationElement aeObj = as.getElementByName(aeName);

            // Get all relations
            ApplicationRelation arList[] = aeObj.getAllRelations();

            int noRel = arList.length;

            // Create a list with names and relations.

```

```

ApplicationElement aeList[] = new ApplicationElement[noRel+1];
aeList[0] = aeObj;
JoinDef joinSeq[] = new JoinDef[noRel];
T_LONGLONG aid = aeObj.getId();
T_LONGLONG toAID;
ApplicationElement aeTo;
int relIndex = 0;
for (int i = 0; i < noRel; i++) {
    // Get the related elements,
    // Only with the correct cardinality will be returned.
    RelationRange relRange = arList[i].getRelationRange();
    if (relRange.max == 1) {
        aeTo = arList[i].getElem2();
        toAID = aeTo.getId();
        if (toAID.low == aid.low) {
            aeList[relIndex+1] = arList[i].getElem1();
            toAID = aeList[relIndex+1].getId();
        } else {
            aeList[relIndex+1] = aeTo;
        }
        joinSeq[relIndex] = new JoinDef();
        joinSeq[relIndex].fromAID = aid;
        joinSeq[relIndex].toAID = toAID;
        joinSeq[relIndex].refName = arList[i].getRelationName();
        if (relRange.min == 0) {
            joinSeq[relIndex].joiningType = JoinType.JTOUTER;
        } else {
            joinSeq[relIndex].joiningType = JoinType.JTDEFAULT;
        }
        relIndex++;
    }
}
noRel = relIndex+1;    // Set the number of used elements.

T_LONGLONG id = new T_LONGLONG();
id.low = 0;
id.high = 0;

// Buildup the query structure
aoq = new QueryStructureExt();

// Report the name of the selected instance and the related instances.
aoq.anuSeq = new SelAIDNameUnitId[noRel];
for (int aeIndex = 0; aeIndex < noRel; aeIndex++) {
    aoq.anuSeq[aeIndex] = new SelAIDNameUnitId();
    aaObj = aeList[aeIndex].getAttributeByBaseName("name");
    aid = aeList[aeIndex].getId();
    aoq.anuSeq[aeIndex].attr = new AIDName(aid, aaObj.getName());
    aoq.anuSeq[aeIndex].unitId = id;
    aoq.anuSeq[aeIndex].aggregate = AggrFunc.NONE;
}

// Set the join sequence
noRel--;
aoq.joinSeq = new JoinDef[noRel];
for (int i = 0; i < noRel; i++) {
    aoq.joinSeq[i] = joinSeq[i];
}

// Condition name like "*"
aaObj = aeObj.getAttributeByBaseName("name");
aoq.condSeq = new SelItem[1];
aoq.condSeq[0] = new SelItem();
SelValueExt sel = new SelValueExt();
sel.attr = new AIDNameUnitId(new AIDName(aeObj.getId(), aaObj.getName()), id);
sel.oper = SelOpcode.LIKE;
sel.value = new TS_Value();

```

```

        sel.value.flag = 15;
        sel.value.u = new TS_Union();
        sel.value.u.stringVal("*");
        aoq.condSeq[0].value(sel);

        // No group by
        aoq.groupBy = new AIDName[0];

        // Order by name
        aoq.orderBy = new SelOrder[1];
        aoq.orderBy[0] = new SelOrder();
        aoq.orderBy[0].attr = new AIDName(aeObj.getId(), aeObj.getName());
        aoq.orderBy[0].ascending = true;

        // Print elapsed time.
        System.out.println ("\nElapsed Time (in ms) for the preparation: " + (System.currentTimeMillis()-mSeconds))

        elemRes = aea.getInstancesExt(aoq, 1000);

        // Print elapsed time.
        System.out.println ("\nElapsed Time (in ms) for the Query: " + (System.currentTimeMillis()-mSeconds));

        if (elemRes.length > 0) {
            showTableResult(elemRes[0]);
        }

        aea = null;
        as = null;

    } catch (AoException aoException) {
        System.err.println("\ntable_query_ext, " +
            aoException.toString() + ":" +
            "\n    " + aoException.reason);
    }
    // Close the active session.
    aoSession.close();

} catch (AoException aoException) {
    System.err.println("\ntable_query_ext, " +
        aoException.toString() + ":" +
        "\n    " + aoException.reason);
}

System.out.println ("\nTotal elapsed Time (in ms): " + (System.currentTimeMillis()-mSeconds));
// Exit the application.
System.exit(0);
}
}

```

4.3.2 deleteInstances of interface ApplElemAccess

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` remove one instance of an application element with the children of the instance element.

Note:

See also **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:[T_LONGLONG](#) aid (in) *The application element Id.*[S_LONGLONG](#) instIds (in) *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);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.UpdateMeasurementTest.testUpdateMeasurement()
```

4.3.3 getAttributeRights of interface ApplElemAccess

Purpose:

Retrieve access control list information for the given application attribute <aid>/<attr_name>.

Return:[ACLSequence](#) aclEntries *The access control list entries of the give application attribute.***Parameter:**[T_LONGLONG](#) aid (in) *The Id of the application element.*[T_STRING](#) attrName (in) *The name of the attribute.***Java Calling Sequence:**

```
ACL[] aclEntries = applElemAccess.getAttributeRights(aid,attrName);
```

Java Example:

```

// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

// Get ApplElemAccess Interface
ApplElemAccess aea = aoSession.getApplElemAccess();

// Get the all application elements
ApplicationElement aeList[] = as.getElements("*");

for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {
    // The access control list.

```



```

ACL [] aclList;
// The Id of the application element
T_LONGLONG aid = aeList[aeIndex].getId();
// The name of the application element.
String aeName = aeList[aeIndex].getName();

// The names of the application attributes, the
// objects are not used at this example.
String aaNames[] = aeList[aeIndex].listAttributes("*");
for (int aaIndex = 0; aaIndex < aaNames.length; aaIndex++) {
    // Get the security rights of each application attribute
    // using the ApplElemAccess interface, an alternative
    // when the attribute objects exist is
    // aclList = aa.getRights();
    aclList = aea.getAttributeRights(aid, aaNames[aaIndex]);
    for (int aclIndex = 0; aclIndex < aclList.length; aclIndex++) {
        System.out.println(aeName + ", " + aaNames[aaIndex] + ", " +
            aclList[aclIndex].usergroupId.low + ", " +
            aclList[aclIndex].rights);
    }
}
}

```

See also:

[getRights](#) of interface [ApplicationAttribute](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.SecurityTest.testRights()

4.3.4 getElementInitialRights of interface ApplElemAccess

Purpose:

Retrieve access control list information of the initial rights for the requested application element <aid>.

Return:

[InitialRightSequence](#) aclEntries *The access control list entries of the given application element for the initial rights.*

Parameter:

[T_LONGLONG](#) aid (in) *The Id of the application element.*

Java Calling Sequence:

```
InitialRight[] aclEntries = applElemAccess.getElementInitialRights(aid);
```

Java Example:

```

// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

```

```
// Get ApplElemAccess Interface
ApplElemAccess aea = aoSession.getApplElemAccess();

/* Get the all application elements. */
ApplicationElement aeList[] = as.getElements("*");

for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {
    InitialRight [] irSeq;          // The Initial rights.

    // Id of the application element.
    T_LONGLONG aid = aeList[aeIndex].getId();
    // Name of the application element
    String aeName = aeList[aeIndex].getName();
    // Get the access control list, using the ApplElemAccess
    // alternative is
    // irSeq = aea.getInitialRights();
    irSeq = aea.getElementInitialRights(aid);
    for (int irIndex = 0; irIndex < irSeq.length; irIndex++) {
        System.out.println(
            aeName +
            irSeq[irIndex].usergroupId.low + ", " +
            irSeq[irIndex].refAid.low + ", " +
            irSeq[irIndex].rights);
    }
}
```

See also:

[getInitialRights](#) of interface [ApplicationElement](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.hqsoft.avalon.SecurityTest.testIniRights()

4.3.5 getElementRights of interface ApplElemAccess

Purpose:

Retrieve access control list information for the requested application element <aid>.

Return:

[ACLSequence](#) aclEntries *The access control list entries of the given application element.*

Parameter:

[T_LONGLONG](#) aid (in) *The Id of the application element.*

Java Calling Sequence:

```
ACL[] aclEntries = applElemAccess.getElementRights(aid);
```

Java Example:

```
// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();
```



```

// Get ApplElemAccess Interface
ApplElemAccess aea = aoSession.getApplElemAccess();

/* Get the all application elements. */
ApplicationElement aeList[] = as.getElements("*");
// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

// Get ApplElemAccess Interface
ApplElemAccess aea = aoSession.getApplElemAccess();

/* Get the all application elements. */
ApplicationElement aeList[] = as.getElements("*");

for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {
    ACL [] aclList;           // The access control list.

    // Id of the application element.
    T_LONGLONG aid = aeList[aeIndex].getId();
    // Name of the application element
    String aeName = aeList[aeIndex].getName();
    // Get the access control list, using the ApplElemAccess
    // alternative is
    // aclList = ae.getRights();
    aclList = aea.getElementRights(aid);
    for (int aclIndex = 0; aclIndex < aclList.length; aclIndex++) {
        System.out.println(
            aeName +
            aclList[aclIndex].usergroupId.low + ", " +
            aclList[aclIndex].rights);
    }
}

```

See also:

[getRights](#) of interface [ApplicationElement](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.SecurityTest.testRights()

4.3.6 getInitialRightReference of interface ApplElemAccess

Purpose:

Get all attribute names (references) which are used to retrieve the Initial Rights.

Return:

[NameSequence](#) refNameList *The names of the references which will be used for the initial rights determination.*

Parameter:

[T_LONGLONG](#) aid (in) *The application element Id.*

Java Calling Sequence:

```
Name[] refNameList = applElemAccess.getInitialRightReference(aid);
```

Java Example:

```
// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

// Get ApplElemAccess Interface
ApplElemAccess aea = aoSession.getApplElemAccess();

// Get all application elements.
ApplicationElement aeList[] = as.getElements("*");

for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {
    // The Id of the application element
    T_LONGLONG aid = aeList[aeIndex].getId();
    // The name of the application element
    String aeName = aeList[aeIndex].getName();

    // The names of the references an alternative is
    // ApplicationRelation aeRels[] = ae.getInitialRightRelation();
    String refName[] = aea.getInitialRightReference(aid);
    for (int refIndex = 0; refIndex < refName.length; refIndex++) {
        System.out.println(aeName + ", " + refName[refIndex]);
    }
}
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.security.ApplModelCheckTest.testApplElemAccessgetInitialRight-
Reference()
```

4.3.7 getInstanceInitialRights of interface ApplElemAccess**Purpose:**

Retrieve access control list information of the initial rights for the requested instance <aid>/<iid>.

Return:

[InitialRightSequence](#) acLEntries *The access control list entries of the given instance for the initial rights.*

Parameter:

[T_LONGLONG](#) aid (in) *The Id of the application element.*

[T_LONGLONG](#) iid (in) *The Id of the instance.*

Java Calling Sequence:

```
InitialRights[] acLEntries = applElemAccess.getInstanceInitialRights(aid,iid);
```



Java Example:

```

// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

// Get ApplElemAccess Interface
ApplElemAccess aea = aoSession.getApplElemAccess();

/* Get the all application elements. */
ApplicationElement aeList[] = as.getElements("*");

for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {
    InitialRight [] irSeq;          // The Initial rights.
    // Id of the application element.
    T_LONGLONG aid = aeList[aeIndex].getId();

    // Name of the application element
    String aeName = aeList[aeIndex].getName();

    // Get the instances of the application element, a query
    // for only the Id of the instances using the ApplElemAccess
    // interface, would do it also, but the query is harder to
    // code and will blowup the example.
    InstanceElement ieObj;
    InstanceElementIterator iei = aeList[aeIndex].getInstances("*");
    if (iei != null) {
        do {
            ieObj = iei.nextOne();
            if (ieObj != null) {
                // Id of the instance,
                T_LONGLONG iid = ieObj.getId();
                // Get the access control list, using the
                // ApplElemAccess interface alternative is
                // irSeq = ieObj.getInitialRights();
                irSeq = aea.getInstanceInitialRights(aid, iid);
                for (int irIndex = 0;
                     irIndex < irSeq.length;
                     irIndex++) {
                    System.out.println(aeName + ", " + iid.low + ", " +
                                         irSeq[irIndex].usergroupId.low + ", " +
                                         irSeq[irIndex].refAid.low + ", " +
                                         irSeq[irIndex].rights);
                }
                // Destroy the instance element, the object is not
                // used anymore at the client.
                ieObj.destroy();
            }
        } while (ieObj != null);
        // Destroy the instance iterator, the object is not
        // used anymore at the client.
        iei.destroy();
    }
} // for all application elements.

```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.SecurityTest.testIniRights()

4.3.8 getInstanceRights of interface ApplElemAccess

Purpose:

Retrieve access control list information for the requested instance <aid>/<iid>.

Return:

[ACLSequence](#) aclEntries *The access control list entries of the given instance.*

Parameter:

[T_LONGLONG](#) aid (in) *The Id of the application element.*

[T_LONGLONG](#) iid (in) *The Id of the instance.*

Java Calling Sequence:

```
ACL[] aclEntries = applElemAccess.getInstanceRights(aid,iid);
```

Java Example:

```
// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

// Get ApplElemAccess Interface
ApplElemAccess aea = aoSession.getApplElemAccess();

// Get the all application elements.
ApplicationElement aeList[] = as.getElements("");

for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {
    ACL [] aclList;           // The access control list.
    int secLevel;           // securtiy level
    // Id of the application element.
    T_LONGLONG aid = aeList[aeIndex].getId();
    // Name of the application element
    String aeName = aeList[aeIndex].getName();

    // Get the security level
    secLevel = aeList[aeIndex].getSecurityLevel();

    if ((secLevel & SecurityLevel.INSTANCE_SECURITY) != 0) {
        // Get the instances of the application element, a query
        // for only the Id of the instances using the ApplElemAccess
        // interface, would do it also, but the query is harder to
        // code and will blowup the example.
        InstanceElement ieObj;
        InstanceElementIterator iei = aeList[aeIndex].getInstances("");
        if (iei != null) {
            do {
                ieObj = iei.nextOne();
                if (ieObj != null) {
                    // Id of the instance,
                    T_LONGLONG iid = ieObj.getId();
                    // Get the access control list, using the
                    // ApplElemAccess interface alternative is
                    // aclList = ieObj.getRights();
                    aclList = aea.getInstanceRights(aid, iid);
                    for (int aclIndex = 0;
                        aclIndex < aclList.length;
                        aclIndex++) {
                        System.out.println(aeName + ", " + iid.low + ", " +
                            aclList[aclIndex].usergroupId.low + ", " +
                            aclList[aclIndex].rights);
                    }
                }
            } while (iei.hasNext());
        }
        // Destroy the instance element, the object is not
```

```

        // used anymore at the client.
        ieObj.destroy();
    }
} while (ieObj != null);
// Destroy the instance iterator, the object is not
// used anymore at the client.
iei.destroy();
}
} // Only when security level is INSTANCE_ELEMENT
} // for all application elements.

```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.3.9 getInstances of interface ApplElemAccess**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 are defined.

The values of the `localcolumn` instances is part of the resultset when the following criteria all fit:

- Only when the attributes `id`, `generation_parameters`, `values`, `flags` are requested, as soon as any other attributes is requested the values are not reported.
- The values of the `localcolumn` must all belong to exactly one instances of `AoMeasurement`
- The server have a limitation of the memory, not the system memory limitation, as soon as the result set will break the limitation an exception will be thrown.
- No iterator for the rest is required.

Note:

Handling of sequence datatypes are not supported at the moment. Read of the values of the `localcolumn` instances only for the same datatype in one call. The `generation_parameters` and `flags` (both with sequence datatypes) will be reported when the `id` and values of the `localcolumns` are also requested.

Return:

[ElemResultSetSequence](#) `elemResultSet` *The result set with the requested attribute values.*

Parameter:

[QueryStructure](#) `aoq` (in) *The query definition.*

[T_LONG](#) `how_many` (in) *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.

Java Calling Sequence:

```
ElemResultSet[] elemResultSet = applElemAccess.getInstance(aoq,how_many);
```

Java Example:

```
import org.asam.ods.AoException;
import org.asam.ods.AoFactory;
import org.asam.ods.AoService;
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.SelOrder;
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 structure.
            ApplicationStructure as = aoSession.getApplicationStructure();

            // Get the appl elem access object
            ApplElemAccess aea = aoSession.getApplElemAccess();

            // Create the query structure for the method getInstance
            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;
    }
    // Set the relation name
    aoq.relName = "";

    // Set the order sequence.
    aoq.orderBy = new SelOrder[0];

    System.out.println("Start call getInstances");
    System.out.println("attribute Id of the instances of AoTest with id < 10 or name like \"ZYK*\" and refe
    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);
}
}

```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.InstanceElementTest.testCheckRelLccMeq()
test.highqsoft.avalon.InstanceElementTest.testInstancesNamesQuery()
test.highqsoft.avalon.MeasurementTest.testMeasurementQuery()
test.highqsoft.avalon.MultiSessionTest.threadInstancesNamesQuery.run()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromSmSelectMea()
test.highqsoft.odsapi.read.InstanceRelationTest.testGetSmInstances()
test.highqsoft.odsapi.read.QueryTest.testInCondition()
test.highqsoft.odsapi.read.QueryTest.testIsNullCondition()
test.highqsoft.odsapi.read.QueryTest.testMultiElements()
test.highqsoft.odsapi.read.QueryTest.testMultiElements2()
....

```

4.3.10 getInstanceExt of interface ApplElemAccess

Purpose:

Perform the Query. This method can be used for a more powerful query compared to the method getInstance 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.

The return is a sequence of the structure ResultSetExt. This sequence will always have the length of 1. This

element in the structure have two fields

- firstElems of type ElemResultSetExtSequence
- restElems of type ElemResultSetExtSeqIterator

firstElems

The firstElems is a sequence of ElemResultSetExt, the length of the sequence corresponds with the number of different application elements requested in the anuSeq of the QueryStructureExt. The structure ElemResultSetExt has the fields aid and values. The aid is the Id of the application element, the values, a sequence of the structure NameValueSeqUnitId are the values of the attributes. For each reported attribute an entry is given in the sequence, the name of the attribute is given in the field valName. The Id of the unit is given in the field unitId, this is the Id of the instance of the application element derived from the base element AoUnit. The values are given in the field value with the type TS_ValueSeq. Each value sequence has the same length, this length is maximum how many the parameter of the method. If no values are found the length of the sequence is 0. The values with the same index in the different sequences of the entry in the NameValueSeqUnitIdSequence belongs to the same instance element.

restElems

The restElems is an iterator for the remaining sequence of ElemResultSetExt. This iterator allows to load the next N values from the ResultSet. The result of the element is the same way stored as at the field firstElems, the next N is the length of the sequence of the field value of the NameValueSeqUnitId. The method nextOne is wrong specified and should not be used, use instead the method nextN with n=1. The method getCount return the total number of found element, also the number already returned in firstElems. The method reset set the iterator the the beginning of the ResultSet, this is the beginning of the firstElems.

The values of the localcolumn instances is part of the resultset when the following criteria all fit:

- Only when the attributes id, generation_parameters, values, flags are requested, as soon as any other attributes is requested the values are not reported.
- The values of the localcolumn must all belong to exactly one instances of AoMeasurement
- The server have a limitation of the memory, not the system memory limitation, as soon as the result set will break the limitation an exception will be thrown.
- No iterator for the rest is required.

Note:

Read of the values of the localcolumn instances only for the same datatype in one call.

Return:

[ResultSetExtSequence](#) resultSet *The result set with the requested attribute values.*

Parameter:

[QueryStructureExt](#) aoq (in) *The query definition.*

T_LONG how_many (in) *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.

Java Calling Sequence:

```
ResultSetExt[] resultSet = applElemAccess.getInstanceExt(aoq,how_many);
```

Since:

ASAM ODS 5.0

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

```
test.highqsoft.avalon.ExtQueryTest.runCatalogElements()
test.highqsoft.avalon.ExtQueryTest.runSimpleQueryExt()
test.highqsoft.avalon.ExtQueryTest.testCatalogElements()
test.highqsoft.avalon.ExtQueryTest.testCoSessCatalogElements()
test.highqsoft.avalon.ExtQueryTest.testCoSessSimpleQueryExt()
test.highqsoft.avalon.ExtQueryTest.testSimpleQueryExt()
test.highqsoft.avalon.MeasurementTest.testManyMeasurement()
test.highqsoft.odsapi.read.ExtendedQueryTest.testAllSeqAttr()
test.highqsoft.odsapi.read.ExtendedQueryTest.testQueryExtRef()
test.highqsoft.odsapi.read.ExtendedQueryTest.testQuerySpecialVersion()
....
```

4.3.11 getRelInst of interface ApplElemAccess

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 defines the unique relation and the target application element.

Return:

S_LONGLONG instIds *The list of the Id of the related instances.*

Parameter:

ElemId elem (in) *Original instance.*

At the RPC API this information was stored in the field elemId of the structure GetInstRefReq and the request AOP_GetInstRef.

Name relName (in) *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);
```

Errors:

```
AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
```

Tested by:

```
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationApplElemAccess()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationApplElemAccess.run()
test.highqsoft.odsapi.CreateInstanceElementTest.createRelationCheck()
test.highqsoft.odsapi.DeleteInstanceElementTest.deleteInstances()
```

4.3.12 getValueMatrix of interface ApplElemAccess**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.

Return:

[ValueMatrix](#) vm *The value matrix*

Parameter:

[ElemId](#) elem (in) *The element id. The aid has to be the application element Id of the measurement or submatrix.*

Java Calling Sequence:

```
ValueMatrix vm = applElemAccess.getValueMatrix(elem);
```

Errors:

```
AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
AO_INVALID_BASETYPE
```

Tested by:

```
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromSmSelectMea()
test.highqsoft.odsapi.read.ApplElemAccessTest.testGetValueMatrix()
test.highqsoft.odsapi.read.ApplElemAccessTest.testGetValueMatrixSecurity()
test.highqsoft.odsapi.CreateMeasurementTest.testAppendValues()
test.highqsoft.odsapi.CreateMeasurementTest.testDeleteValues()
```

4.3.13 insertInstances of interface ApplElemAccess

Purpose:

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.

Note:

See also **Transaction handling in ODS API.**(p. 14)

Return:

[ElemIdSequence](#) elemIds *List with the Id's of the new created instances.*

Parameter:

[AIDNameValueSeqUnitIdSequence](#) val (in) *The sequence of attributes and their values.*

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:

```
ElemIdSequence elemIds = applElemAccess.insertInstances(val);
```

Java Example:

```
/**
 * ASAM ODS Example: Using ApplElemAccess, the insertInstances method.
 *
 */

import org.asam.ods.AoException;
import org.asam.ods.AoFactory;
import org.asam.ods.AoService;
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.BaseAttribute;
import org.asam.ods.DataType;
import org.asam.ods.AIDName;
import org.asam.ods.AIDNameValueSeqUnitId;
import org.asam.ods.NameValueSeqUnitId;
import org.asam.ods.ElemId;
import org.asam.ods.TS_ValueSeq;
import org.asam.ods.TS_UnionSeq;
import org.asam.ods.T_LONGLONG;
import org.asam.ods.AttrResultSet;
import org.asam.ods.QueryStructure;
import org.asam.ods.AIDNameUnitId;
import org.asam.ods.AIDNameValueUnitId;
```

```

import org.asam.ods.ElemResultSet;
import org.asam.ods.SelOpcode;
import org.asam.ods.SelOperator;
import org.asam.ods.SelOrder;
import org.asam.ods.SelValue;
import org.asam.ods.TS_Value;
import org.asam.ods.TS_Union;

import com.highqsoft.ods.AoServiceFactory;

import com.highqsoft.odsx.OdsxHelper;

public class InsertApplElemAccess {

    // Java application entry.
    public static void main(String[] args) {

        // Say in which example we are.
        System.out.println("\nExample InsertApplElemAccess...");
        System.out.println("-----");

        // Timestamp for performance check.
        long mSeconds = System.currentTimeMillis();

        try {

            // Establish a session to the service.
            AoSession aoSession = ExamplesConnect.connectService(args);

            try {

                // Get the application structure.
                ApplicationStructure as = aoSession.getApplicationStructure();

                // Get the appl elem access object
                ApplElemAccess aeaObj = aoSession.getApplElemAccess();

                // The application attribute.
                ApplicationAttribute aaObj;

                // Selectvalue of the Id
                T_LONGLONG iid;

                // Number of instances
                int      noInst = 2;
                String   strPrefix = "Str";
                // Query on the application element AoTest

                ApplicationElement ae[];
                ae = as.getElementsByBaseType ("AoTest");
                ApplicationElement aeObj = null;

                if (ae.length > 0)
                {
                    aeObj = ae[0];

                    /* Get the Id of the application element */
                    T_LONGLONG aid = aeObj.getId();

                    /* Get the attributes of the application element */
                    ApplicationAttribute aaList[] = aeObj.getAttributes("");

                    /* Create the list for the attributes leave out the attribute Id */
                    AIDNameValueSeqUnitId val[] = new AIDNameValueSeqUnitId[aaList.length-1];

                    BaseAttribute baObj;
                    int valIndex = 0;

```

```

int skip;
for (int aaIndex = 0; aaIndex < aaList.length; aaIndex++) {
    skip = 0;
    aaObj = aaList[aaIndex];
    baObj = aaObj.getBaseAttribute();
    if (baObj != null) {
        if (baObj.getName().compareToIgnoreCase("id") == 0) {
            skip = 1;
        }
    }
}

if (skip == 0) {
    val[valIndex] = new AIDNameValueSeqUnitId();
    val[valIndex].attr = new AIDName();
    val[valIndex].attr.aid = aid;
    val[valIndex].attr.aaName = aaObj.getName();
    val[valIndex].unitId = aaObj.getUnit();
    val[valIndex].values = new TS_ValueSeq();
    val[valIndex].values.flag = new short[noInst];
    val[valIndex].values.u = new TS_UnionSeq();
    DataType dt = aaObj.getDataType();
    switch (dt.value()) {
        case DataType._DT_STRING:
        {
            String s[] = new String[noInst];
            for (int i = 0; i < noInst; i++) {
                s[i] = strPrefix + i;
                val[valIndex].values.flag[i] = 15;
            }
            val[valIndex].values.u.stringVal(s);
            break;
        }
        case DataType._DT_DATE:
        {
            String s[] = new String[noInst];
            for (int i = 0; i < noInst; i++) {
                s[i] = "20031015150000";
                val[valIndex].values.flag[i] = 15;
            }
            val[valIndex].values.u.dateVal(s);
            break;
        }
        case DataType._DT_LONG:
        {
            int s[] = new int[noInst];
            for (int i = 0; i < noInst; i++) {
                s[i] = i;
                val[valIndex].values.flag[i] = 15;
            }
            val[valIndex].values.u.longVal(s);
            break;
        }
        case DataType._DT_FLOAT:
        {
            float s[] = new float[noInst];
            for (int i = 0; i < noInst; i++) {
                s[i] = (float)i;
                val[valIndex].values.flag[i] = 15;
            }
            val[valIndex].values.u.floatVal(s);
            break;
        }
        default:
        {
            for (int i = 0; i < noInst; i++) {
                val[valIndex].values.flag[i] = 0;
            }
        }
    }
}

```

```

        break;
    }
    }
    valIndex++;
} // not skip
} // for all attributes

aoSession.startTransaction();
ElemId elemIds[] = aeaObj.insertInstances(val);
for (int i = 0; i < elemIds.length; i++) {
    System.out.println("aid = " + elemIds[i].aid.low + " iid = " + elemIds[i].iid.low);
}
aoSession.commitTransaction();

// Print elapsed time.
System.out.println ("\nElapsed Time (in ms): " + (System.currentTimeMillis()-mSeconds));
mSeconds = System.currentTimeMillis();

}
} catch (AoException aoException) {
    System.err.println("\nInsertApplElemAccess, " +
        aoException.toString() + ":" +
        "\n    " + aoException.reason);
    aoException.printStackTrace();
}

// Close the active session.
aoSession.close();

} catch (AoException aoException) {
    System.err.println("\nInsertApplElemAccess, " +
        aoException.toString() + ":" +
        "\n    " + aoException.reason);
}

// Print elapsed time.
System.out.println ("\nElapsed Time (in ms): " + (System.currentTimeMillis()-mSeconds));

// Exit the application.
System.exit(0);

}

}

```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.odsapi.security.InstanceCheckTest.testSubTestRights()
test.highqsoft.odsapi.TransactionTest.testCommitCheckInst()
test.highqsoft.odsapi.WriteInstanceElementTest.testDuplicateNameAEA()
test.highqsoft.odsapi.WriteInstanceElementTest.testInvalidRequest()
test.highqsoft.odsapi.WriteInstanceElementTest.testLongNameAEA()

4.3.14 setAttributeRights of interface ApplElemAccess

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.

Note:

See also **Transaction handling in ODS API.**(p. 14)

Return:

void

Parameter:

T_LONGLONG aid (in) *The Id of the application element.*

T_STRING attrName (in) *The name of the attribute.*

T_LONGLONG usergroupId (in) *The usergroup to set the rights for.*

T_LONG rights (in) *The new right for the usergroup.*

RightsSet set (in) *What to do with the new right.*

Java Calling Sequence:

```
applElemAccess.setAttributeRights(aid,attrName,usergroupId,rights,set);
```

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
AO_TRANSACTION_NOT_ACTIVE

4.3.15 setElementInitialRights of interface ApplElemAccess

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.

Return:

void

Parameter:**T_LONGLONG** aid (in) *The Id of the application element.***T_LONGLONG** usergroupId (in) *The usergroup to set the initial rights for.***T_LONG** rights (in) *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.***T_LONGLONG** refAid (in) *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.***RightsSet** set (in) *What to do with the new initial rights.***Java Calling Sequence:**`applElemAccess.setElementInitialRights(aid,usergroupId,rights,refAid,set);`**Errors:**

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:`test.highqsoft.odsapi.security.InstanceCheckTest.testCurrentRights()`**4.3.16 setElementRights of interface ApplElemAccess****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.

Note:See also **Transaction handling in ODS API.**(p. 14)**Return:**

void

Parameter:**T_LONGLONG** aid (in) *The Id of the application element.*

T_LONGLONG usergroupId (in) *The usergroup to set the rights for.*

T_LONG rights (in) *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.*

RightsSet set (in) *What to do with the new right.*

Java Calling Sequence:

```
applElemAccess.setElementRights(aid,usergroupId,rights,set);
```

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

4.3.17 setInitialRightReference of interface ApplElemAccess

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.

Return:

void

Parameter:

T_LONGLONG aid (in) *The application element Id.*

T_STRING refName (in) *The name of the reference.*

RightsSet set (in) *What to do with the given reference.*

Java Calling Sequence:

```
applElemAccess.setInitialRightReference(aid,refName,set);
```

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
AO_TRANSACTION_NOT_ACTIVE

4.3.18 setInstanceInitialRights of interface ApplElemAccess

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.

Note:

See also **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

T_LONGLONG aid (in) *The Id of the application element.*

S_LONGLONG instIds (in) *The sequence of instance Id's.*

T_LONGLONG usergroupId (in) *The usergroup to set the initial rights for.*

T_LONG rights (in) *The new initial right for the usergroup.*

T_LONGLONG refAid (in) *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.*

RightsSet set (in) *What to do with the new initial rights.*

Java Calling Sequence:

```
applElemAccess.setInstanceInitialRights(aid,instIds,usergroupId,rights,refAid,set);
```

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

4.3.19 setInstanceRights of interface ApplElemAccess

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.

Note:

See also **Transaction handling in ODS API.**(p. 14)

Return:

void

Parameter:

T_LONGLONG aid (in) *The Id of the application element.*

S_LONGLONG instIds (in) *The sequence of instance Id's.*

T_LONGLONG usergroupId (in) *The usergroup to set the rights for.*

T_LONG rights (in) *The new right for the usergroup.*

RightsSet set (in) *What to do with the new right.*

Java Calling Sequence:

```
applElemAccess.setInstanceRights(aid,instIds,usergroupId,rights,set);
```

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

```
test.highqsoft.odsapi.security.InstanceCheckTest.testCurrentRights()
```

4.3.20 setRelInst of interface ApplElemAccess

Purpose:

Set the instance reference.

Note:

See also **Transaction handling in ODS API.**(p. 14)

Return:

void

Parameter:

ElemId elem (in) *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.



Name relName (in) *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.

S_LONGLONG instIds (in) *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 possible to set more than one relation.

SetType type (in) *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);
```

Errors:

```
AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
```

Tested by:

```
test.highqsoft.odsapi.CreateInstanceElementTest.testN_MCreate()
test.highqsoft.odsapi.DeleteInstanceElementTest.deleteInstances()
```

4.3.21 updateInstances of interface ApplElemAccess

Purpose:

Update the one or more attributes of one or more instance elements. It's 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. The server will update the values of the localcolumn instances when the following criteria all fit:

- Only when the attributes id, generation_parameters, values, flags are given, as soon as any other attributes is given an exception is thrown.
- The values of the localcolumn must all belong to exactly one instances of AoMeasurement
- The server have a limitation of the memory, not the system memory limitation, as soon as the given values breaks the limitation an exception will be thrown.

Note:

See also **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

[AIDNameValueSeqUnitIdSequence](#) val (in) *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);
```

Java Example:

```
/**
 * ASAM ODS Example: Using ApplElemAccess, the updateInstanes method.
 *
 */

import org.asam.ods.AoException;
import org.asam.ods.AoFactory;
import org.asam.ods.AoService;
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.BaseAttribute;
import org.asam.ods.DataType;
import org.asam.ods.AIDName;
import org.asam.ods.AIDNameValueSeqUnitId;
import org.asam.ods.NameValueSeqUnitId;
import org.asam.ods.ElemId;
import org.asam.ods.TS_ValueSeq;
import org.asam.ods.TS_UnionSeq;
import org.asam.ods.T_LONGLONG;
import org.asam.ods.AttrResultSet;
import org.asam.ods.QueryStructure;
import org.asam.ods.AIDNameUnitId;
import org.asam.ods.AIDNameValueUnitId;
import org.asam.ods.ElemResultSet;
import org.asam.ods.SelOpcode;
import org.asam.ods.SelOperator;
import org.asam.ods.SelOrder;
import org.asam.ods.SelValue;
import org.asam.ods.TS_Value;
import org.asam.ods.TS_Union;

import com.highqsoft.ods.AoServiceFactory;

import com.highqsoft.odsx.OdsxHelper;

public class UpdateApplElemAccess {

    public static void showElemResultSet (ElemResultSet elemRes[]) throws AoException
    {
        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);
                System.out.println(OdsxHelper.ts_valueSeqToString(elemRes[i].attrValues[j].attrValues.value));
            }
        }
    }

    // Java application entry.
    public static void main(String[] args) {

        // Say in which example we are.
```

```

System.out.println("\nExample UpdateApplElemAccess...");
System.out.println("-----");

// Timestamp for performance check.
long mSeconds = System.currentTimeMillis();

try {

    // Establish a session to the service.
    AoSession aoSession = ExamplesConnect.connectService(args);

    try {

        // Get the application structure.
        ApplicationStructure as = aoSession.getApplicationStructure();

        // Get the appl elem access object
        ApplElemAccess aeaObj = aoSession.getApplElemAccess();

        // The application attribute.
        ApplicationAttribute aaObj;

        // Selectvalue of the Id
        T_LONGLONG iid;

        String    strPrefix = "Str";

        // Query on the application element AoTest

        ApplicationElement ae[];
        ae = as.getElementsByBaseType ("AoTest");
        ApplicationElement aeObj = null;

        if (ae.length > 0)
        {
            aeObj = ae[0];

            /* Get the Id of the application element */
            T_LONGLONG aid = aeObj.getId();

            /* Get the attributes of the application element */
            ApplicationAttribute aaList[] = aeObj.getAttributes("*");

            /* Create the list for the attributes leave out the attribute Id */
            AIDNameValueSeqUnitId val[];

            /* Query the Id and Version from the just inserted Instances using the name with the strPrefix */
            QueryStructure aoq = new QueryStructure();

            aaObj = aeObj.getAttributeByBaseName("version");
            if (aaObj != null) {
                // Report version and Id attribute
                aoq.anuSeq = new AIDNameUnitId[2];
                aoq.anuSeq[0] = new AIDNameUnitId();
                aoq.anuSeq[0].attr = new AIDName();
                aoq.anuSeq[0].attr.aid = aeObj.getId();
                aoq.anuSeq[0].attr.aaName = aaObj.getName();
                aoq.anuSeq[0].unitId = new T_LONGLONG(0,0);

                aaObj = aeObj.getAttributeByBaseName("id");
                aoq.anuSeq[1] = new AIDNameUnitId();
                aoq.anuSeq[1].attr = new AIDName();
                aoq.anuSeq[1].attr.aid = aeObj.getId();
                aoq.anuSeq[1].attr.aaName = aaObj.getName();
                aoq.anuSeq[1].unitId = new T_LONGLONG(0,0);

                // Condition name like strPrefix*

```

```

aaObj = aeObj.getAttributeByBaseName("name");
aoq.condSeq = new SelValue[1];
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].attr.unitId = new T_LONGLONG(0,0);
aoq.condSeq[0].attr.values = new TS_Value();
aoq.condSeq[0].attr.values.flag = 15;
aoq.condSeq[0].attr.values.u = new TS_Union();
aoq.condSeq[0].attr.values.u.longVal(0); // Dummy

// The inset
aoq.condSeq[0].oper = SelOpcode.LIKE;

aoq.condSeq[0].value = new TS_Value();
aoq.condSeq[0].value.flag = 15;
aoq.condSeq[0].value.u = new TS_Union();
aoq.condSeq[0].value.u.stringVal(strPrefix+"*");

// Initialize the not used fields

aoq.operSeq = new SelOperator[0];
aoq.orderBy = new SelOrder[0];
aoq.relInst = new ElemId();
aoq.relInst.aid = new T_LONGLONG(0,0);
aoq.relInst.iid = new T_LONGLONG(0,0);
aoq.relName = "";

ElemResultSet elemRes[] = aeaObj.getInstance(aoq, 1000);

// Print elapsed time.
System.out.println ("\nElapsed Time (in ms): " + (System.currentTimeMillis()-mSeconds));
if (elemRes != null) {
    showElemResultSet(elemRes);
    mSeconds = System.currentTimeMillis();
    /* Now are the Id's and the versions loaded create the val for the update of the version. */

    // Get the version attribute to compare the name
    aaObj = aeObj.getAttributeByBaseName("version");
    String aaVersionName = aaObj.getName();

    NameValueSeqUnitId attrRes;
    val = new AIDNameValueSeqUnitId[2];
    for (int i = 0; i < 2; i++) {
        val[i] = new AIDNameValueSeqUnitId();
        val[i].attr = new AIDName();
        val[i].attr.aid = aeObj.getId();
        attrRes = elemRes[0].attrValues[i].attrValues;
        val[i].attr.aaName = attrRes.valName;
        val[i].unitId = attrRes.unitId;
        if (aaVersionName.compareTo(attrRes.valName) == 0) {
            /* Overwrite the version */
            String v[] = attrRes.value.u.stringVal();
            val[i].values = new TS_ValueSeq();
            val[i].values.flag = new short[v.length];
            for (int j = 0; j < v.length; j++) {
                v[j] = "V_" + j;
                val[i].values.flag[j] = 15;
            }
            val[i].values.u = new TS_UnionSeq();
            val[i].values.u.stringVal(v);
        } else {
            /* Copy the result from the query result, this is
             * the Id of the instances.
             */

```

```

        val[i].values = attrRes.value;
    }
}

aoSession.startTransaction();
// Update the instances
aeaObj.updateInstances(val);
aoSession.commitTransaction();

// Print elapsed time.
System.out.println ("\nElapsed Time (in ms) fro update: " + (System.currentTimeMillis()-mSeconds));
mSeconds = System.currentTimeMillis();
}

// Same query again
elemRes = aeaObj.getInstances(aoq, 1000);

// Print elapsed time.
System.out.println ("\nElapsed Time (in ms): " + (System.currentTimeMillis()-mSeconds));
if (elemRes != null) {
    showElemResultSet(elemRes);
}

} /* No version attribute */
}
} catch (AoException aoException) {
    System.err.println("\nUpdateApplElemAccess, " +
        aoException.toString() + ":" +
        "\n    " + aoException.reason);
    aoException.printStackTrace();
}

// Close the active session.
aoSession.close();

} catch (AoException aoException) {
    System.err.println("\nUpdateApplElemAccess, " +
        aoException.toString() + ":" +
        "\n    " + aoException.reason);
}

// Print elapsed time.
System.out.println ("\nElapsed Time (in ms): " + (System.currentTimeMillis()-mSeconds));

// Exit the application.
System.exit(0);
}
}

```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.odsapi.security.InstanceCheckTest.testSubTestRights()
 test.highqsoft.odsapi.UpdateInstanceElementTest.testUpdateDate()
 test.highqsoft.odsapi.WriteInstanceElementTest.testUpdateLocalColumn()
 test.highqsoft.odsapi.WriteInstanceElementTest.testUpdateRead()

4.4 ApplicationAttribute

Package:

org.asam.ods

Description:

The ASAM ODS application attribute interface.

4.4.1 getApplicationElement of interface ApplicationAttribute

Purpose:

Return the application element to which the attribute belongs.

Return:

[ApplicationElement](#) applElem *The application element of the attribute.*

Parameter:

None.

Java Calling Sequence:

```
ApplicationElement applElem = applAttr.getApplicationElement();
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.4.2 getBaseAttribute of interface ApplicationAttribute

Purpose:

Get the base attribute of the application attribute.

Return:

[BaseAttribute](#) baseAttr *The base attribute of the application attribute. A 'null' is returned if the application attribute has no base attribute.*

Parameter:

None.

Java Calling Sequence:

```
BaseAttribute baseAttr = applAttr.getBaseAttribute();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)



Tested by:

```

test.highqsoft.odsapi.changemodel.ApplElemCreateTest.createNewInstance()
test.highqsoft.odsapi.ApplModelCreateTest.testCheckElement()
test.highqsoft.odsapi.AthosTestCase.getWriteableAttributes()
test.highqsoft.odsapi.InstanceAttributeTest.testSetValue()
test.highqsoft.odsapi.InstanceAttributeTest.testSetValueSeq()
test.highqsoft.odsapi.InstanceAttributeTest.testSingleInstanceAttributeDatatype()
test.highqsoft.odsapi.InstanceAttributeTest.testAddInstanceAttribute()
test.highqsoft.odsapi.InstanceAttributeTest.testManyInstanceAttributeDatatype()
test.highqsoft.odsapi.UpdateInstanceElementTest.testUpdateExtRef()
test.highqsoft.odsapi.WriteInstanceElementTest.CompareInstancesWithChild()
....

```

4.4.3 getDataType of interface ApplicationAttribute**Purpose:**

Get the data type of the application attribute.

Return:

[DataType](#) aaDataType *The data type of the application attribute.*

Parameter:

None.

Java Calling Sequence:

```
DataType aaDataType = applAttr.getDataType();
```

Errors:

```

AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

```

Tested by:

```

test.highqsoft.odsapi.read.AsamPathTest.testCreateAsamPathSpaces()
test.highqsoft.odsapi.CreateInstanceTestCase.createLcInst()
test.highqsoft.odsapi.CreateInstanceTestCase.createMeqInst()
test.highqsoft.odsapi.CreateMeasurementTest.getSubMatrixByName()
test.highqsoft.odsapi.UpdateMeasurementTest.createMeasurementBody()
test.highqsoft.odsapi.UpdateMeasurementTest.testUpdateMeasurement()
test.highqsoft.odsapi.WriteInstanceElementTest.testWriteRelAttr()

```

4.4.4 getEnumerationDefinition of interface ApplicationAttribute**Purpose:**

Get the definition of the enumeration.

Return:

[EnumerationDefinition](#) enumDef *The ASAM ODS enumeration.*

Parameter:

None.

Java Calling Sequence:

```
EnumerationDefinition enumDef = aaObj.getEnumerationDefinition();
```

Errors:

```
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
AO_INVALID_DATATYPE
```

Tested by:

```
test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadTestEnumeration-
Definition.run()
test.highqsoft.avalon.ApplicationStructureTest.testEnumerationDefinition()
test.highqsoft.odsapi.EnumerationTest.testReadEnumInstance()
test.highqsoft.odsapi.EnumerationTest.testWriteEnumInstance()
```

4.4.5 getLength of interface ApplicationAttribute**Purpose:**

Get the maximum allowed length of the value of the application attribute.

Return:

T_LONG aaLength *The maximum allowed length of the application attribute.*

Parameter:

None.

Java Calling Sequence:

```
T_LONG aaLength = applAttr.getLength();
```

Java Example:

```
// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

// Get ApplElemAccess Interface
ApplElemAccess aea = aoSession.getApplElemAccess();

// Get the all application elements
ApplicationElement aeList[] = as.getElements("*");

for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {
    // The name of the application element.
    String aeName = aeList[aeIndex].getName();

    // The application attributes.
    ApplicationAttribute aaList[] = aeList[aeIndex].getAttributes("*");
    for (int aaIndex = 0; aaIndex < aaList.length; aaIndex++) {
        // Get the name of the application attribute
        String aaName = aaList[aaIndex].getName();
        System.out.println(aeName + ", " + aaName + " = " +
            aaList[aaIndex].getLength());
    }
}
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.hiqsoft.odsapi.ApplModelCreateTest.testSetLength()

4.4.6 getName of interface ApplicationAttribute**Purpose:**

Get the name of the application attribute.

Return:

[Name](#) aaName *The name of the application attribute.*

Parameter:

None.

Java Calling Sequence:

```
Name aaName = applAttr.getName();
```

Java Example:

```
// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

// Get ApplElemAccess Interface
ApplElemAccess aea = aoSession.getApplElemAccess();

// Get the all application elements
ApplicationElement aeList[] = as.getElements("*");

for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {
    // The name of the application element.
    String aeName = aeList[aeIndex].getName();

    // The application attributes.
    ApplicationAttribute aaList[] = aeList[aeIndex].getAttributes("*");
    for (int aaIndex = 0; aaIndex < aaList.length; aaIndex++) {
        // Get the name of the application attribute
        String aaName = aaList[aaIndex].getName();
        System.out.println(aeName + ", " + aaName);
    }
}
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.hiqsoft.avalon.ExtQueryTest.runCatalogElements()

```

test.highqsoft.avalon.ExtQueryTest.runSimpleQueryExt()
test.highqsoft.avalon.ExtQueryTest.testCatalogElements()
test.highqsoft.avalon.ExtQueryTest.testCoSessCatalogElements()
test.highqsoft.avalon.ExtQueryTest.testCoSessSimpleQueryExt()
test.highqsoft.avalon.ExtQueryTest.testSimpleQueryExt()
test.highqsoft.avalon.InstanceElementTest.testCheckRelLccMeq()
test.highqsoft.avalon.InstanceElementTest.testInstancesNamesQuery()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationApplElemAccess()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelations()
....

```

4.4.7 getRights of interface ApplicationAttribute

Purpose:

Retrieve access control list information of the given object.

Return:

[ACLSequence](#) acListEntries *The access control list entries of the given application element.*

Parameter:

None.

Java Calling Sequence:

```
ACL[] acListEntries = applAttr.getRights();
```

Java Example:

```

// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

// Get ApplElemAccess Interface
ApplElemAccess aea = aoSession.getApplElemAccess();

// Get the all application elements
ApplicationElement aeList[] = as.getElements("*");

for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {
    // The access control list.
    ACL[] acList;
    // The name of the application element.
    String aeName = aeList[aeIndex].getName();

    // The application attributes.
    ApplicationAttribute aaList[] = aeList[aeIndex].getAttributes("*");
    for (int aaIndex = 0; aaIndex < aaList.length; aaIndex++) {
        // Get the security rights of each application attribute
        acList = aaList[aaIndex].getRights();
        String aaName = aaList[aaIndex].getName();
        for (int aclIndex = 0; aclIndex < acList.length; aclIndex++) {
            System.out.println(aeName + ", " + aaName + ", " +
                acList[aclIndex].usergroupId.low + ", " +
                acList[aclIndex].rights);
        }
    }
}

```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)



AO_IMPLEMENTATION_PROBLEM
 AO_NOT_IMPLEMENTED
 AO_NO_MEMORY
 AO_SESSION_NOT_ACTIVE

4.4.8 getUnit of interface ApplicationAttribute

Purpose:

Get the unit Id of the application attribute. The unit Id is only valid for the current server.

Return:

T_LONGLONG aaUnit *The unit Id of the application attribute.*

Parameter:

None.

Java Calling Sequence:

```
T_LONGLONG aaUnit = applAttr.getUnit();
```

Java Example:

```
// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

// Get ApplElemAccess Interface
ApplElemAccess aea = aoSession.getApplElemAccess();

// Get the all application elements
ApplicationElement aeList[] = as.getElements("*");

for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {
    // The name of the application element.
    String aeName = aeList[aeIndex].getName();

    // The application attributes.
    ApplicationAttribute aaList[] = aeList[aeIndex].getAttributes("*");
    for (int aaIndex = 0; aaIndex < aaList.length; aaIndex++) {
        // Get the name of the application attribute
        String aaName = aaList[aaIndex].getName();
        System.out.println(aeName + ", " + aaName + " [" +
            aaList[aaIndex].getUnit() + "]");
    }
}
```

Errors:

AO_CONNECTION_LOST
 AO_IMPLEMENTATION_PROBLEM
 AO_NOT_IMPLEMENTED
 AO_NO_MEMORY
 AO_SESSION_NOT_ACTIVE

Tested by:

```
test.highqsoft.odsapi.read.UnitTest.testGetInstancesExt AnuSeq()
```

4.4.9 hasUnit of interface ApplicationAttribute

Purpose:

Has the attribute an unit. If this flag is set, all the attributes of the instances derived from this attribute will has an unit.

Return:

[T_BOOLEAN](#) hasUnit *The flag if the attribute has an unit.*

Parameter:

None.

Java Calling Sequence:

```
Boolean hasUnit = aaObj.hasUnit();
```

Since:

ASAM ODS 5.0

See also:

[withUnit](#) of interface [ApplicationAttribute](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.4.10 hasValueFlag of interface ApplicationAttribute

Purpose:

Has the attribute a value flag. If this flag is set, all the attributes of the instances derived from this attribute will has a value flag. If this flag is not set the flag in the TS_Value structure can be ignored.

Note:

Not yet implemented. Deprecated in ASAM ODS 5.2

Return:

[T_BOOLEAN](#) hasValueFlag *The flag if the attribute has a value flag.*

Parameter:

None.

Java Calling Sequence:

```
Boolean hasValueFlag = aaObj.hasValueFlag();
```

Since:

ASAM ODS 5.0

See also:

[withValueFlag](#) of interface [ApplicationAttribute](#)



Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

4.4.11 isAutogenerated of interface ApplicationAttribute**Purpose:**

Get the autogenerated flag of the application attribute.

Return:

T_BOOLEAN isAutogenerated *The autogenerated flag of the application attribute.*

Parameter:

None.

Java Calling Sequence:

```
T_BOOLEAN isAutogenerated = applAttr.IsAutogenerated();
```

Errors:

AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

4.4.12 isObligatory of interface ApplicationAttribute**Purpose:**

Get the obligatory flag of the application attribute.

Return:

T_BOOLEAN aaIsObligatory *The obligatory flag of the application attribute.*

Parameter:

None.

Java Calling Sequence:

```
T_BOOLEAN isObligatory = applAttr.isObligatory();
```

Errors:

AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

4.4.13 isUnique of interface ApplicationAttribute

Purpose:

Get the unique flag of the application attribute.

Return:

T_BOOLEAN aaIsUnique *The unique flag of the application attribute.*

Parameter:

None.

Java Calling Sequence:

```
T_BOOLEAN isUnique = applAttr.isUnique();
```

Errors:

AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

```
test.highqsoft.odsapi.CreateInstanceTestCase.createInstances()
```

4.4.14 setBaseAttribute of interface ApplicationAttribute

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 instantiated 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.

Note:

It is only allowed to modify an attribute of an application element when there are no instance stored at the application element. For the transaction handling see section **Transaction handling in ODS API.**(p. 14)

Return:

void

Parameter:[BaseAttribute](#) baseAttr (in) *The base attribute.***Java Calling Sequence:**`applAttr.setBaseAttribute(baseAttr);`**Errors:**

[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](#)

Tested by:

test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testAddAttribute()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateElement()
test.highqsoft.odsapi.ApplModelCreateTest.testSetBaseAttr()

4.4.15 setDataType of interface ApplicationAttribute**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 instantiated attributes.

Note:

It is only allowed to modify an attribute of an application element when there are no instance stored at the application element. For the transaction handling see section **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

[DataType](#) aaDataType (in) *The data type.*

Java Calling Sequence:

```
applAttr.setDataType(aaDataType);
```

Errors:

[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](#)

Tested by:

```
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testAddAttribute()
test.highqsoft.odsapi.ApplModelCreateTest.testCreateAttr()
test.highqsoft.odsapi.ApplModelCreateTest.testSetUnit()
```

4.4.16 setEnumerationDefinition of interface ApplicationAttribute

Purpose:

Set the definition of the enumeration. This method modifies the application model, only the superuser can use this method.

The AoException [AO_INVALID_DATATYPE](#) is thrown when the datatype of the attribute is not [DT_ENUM](#).

Note:

It is only allowed to modify an attribute of an application element when there are no instance stored at the application element. For the transaction handling see section **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

[EnumerationDefinition](#) enumDef (in) *The new enumeration definition.*

Java Calling Sequence:

```
aaObj.setEnumerationDefinition(enumDef);
```

Since:

ASAM ODS 5.0

Errors:

[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_INVALID_DATATYPE](#)

4.4.17 setIsAutogenerated of interface ApplicationAttribute

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.

Note:

It is only allowed to modify an attribute of an application element when there are no instance stored at the application element. For the transaction handling see section **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

[T_BOOLEAN](#) isAutogenerated (in) *The autogenerate flag.*

Java Calling Sequence:

```
applAttr.setIsAutogenerated(isAutogenerated);
```

Errors:

[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](#)

Tested by:

```
test.hiqsoft.odsapi.ApplModelCreateTest.testSetUnit()
```

4.4.18 setIsObligatory of interface ApplicationAttribute

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.

Note:

It is only allowed to modify an attribute of an application element when there are no instance stored at the application element. For the transaction handling see section **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

`T_BOOLEAN` aaIsObligatory (in) *The obligatory flag.*

Java Calling Sequence:

```
applAttr.setIsObligatory(aaIsObligatory);
```

Errors:

`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`

Tested by:

```
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testModAttrObligatory()
test.highqsoft.odsapi.ApplModelCreateTest.testSetUnit()
```

4.4.19 setIsUnique of interface ApplicationAttribute

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 `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.

Note:

It is only allowed to modify an attribute of an application element when there are no instance stored at the application element. The check of the uniqueness of the instance attributes values is not implemented.

For the transaction handling see section **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

T_BOOLEAN aaIsUnique (in) *The unique flag.*

Java Calling Sequence:

```
applAttr.setIsUnique(aaIsUnique);
```

Errors:

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

Tested by:

```
test.highqsoft.odsapi.ApplModelCreateTest.testSetUnit()
```

4.4.20 setLength of interface ApplicationAttribute

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.

Note:

It is only allowed to modify an attribute of an application element when there are no instance stored at the application element. For the transaction handling see section **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

[T_LONG](#) aaLength (in) *The maximum attribute length.*

Java Calling Sequence:

```
applAttr.setLength(aaLength);
```

Errors:

[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](#)

Tested by:

```
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testAddAttribute()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testModAttrLength()
test.highqsoft.odsapi.ApplModelCreateTest.testCreateAttr()
test.highqsoft.odsapi.ApplModelCreateTest.testSetLength()
```

4.4.21 setName of interface ApplicationAttribute

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. Current server implementations typically restrict it to 30 characters.

Note:

It is only allowed to modify an attribute of an application element when there are no instance stored at the application element. For the transaction handling see section **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

Name aaName (in) *The application attribute name.*

Java Calling Sequence:

```
applAttr.setName(aaName);
```

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_DUPLICATE_NAME
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

```
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testAddAttribute()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testRenameAttribute()
test.highqsoft.odsapi.ApplModelCreateTest.testCreateAttr()
test.highqsoft.odsapi.ApplModelCreateTest.testSetBaseAttr()
test.highqsoft.odsapi.ApplModelCreateTest.testSetUnit()
```

4.4.22 setRights of interface ApplicationAttribute**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.

Return:

void

Parameter:

InstanceElement usergroup (in) *The usergroup for which the rights will be modified.*

T_LONG rights (in) *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.*

RightsSet set (in) *What to do with the new right.*

Java Calling Sequence:

```
applAttr.setRights(usergroup, rights, set);
```

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
AO_TRANSACTION_NOT_ACTIVE

Tested by:

```
test.highqsoft.odsapi.security.AttributeCheckTest.testSetAttributeRights()
```

4.4.23 setUnit of interface ApplicationAttribute

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.

Note:

It is only allowed to modify an attribute of an application element when there are no instance stored at the application element. No check and modification of the unit ids of the instance attributes are implemented. For the transaction handling see section **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

T_LONGLONG aaUnit (in) *The unit Id.*

Java Calling Sequence:

```
applAttr.setUnit(aaUnit);
```

Errors:

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

Tested by:

```
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testAddAttribute()  
test.highqsoft.odsapi.ApplModelCreateTest.testSetUnit()
```

4.4.24 withUnit of interface ApplicationAttribute**Purpose:**

Set whether the attribute becomes an unit or not. A call to the method setUnit() will automatically set the withUnit(TRUE).

Note:

It is only allowed to modify an attribute of an application element when there are no instance stored at the application element.

For the transaction handling see section **Transaction handling in ODS API**.(p. 14)

Not yet implemented.

Return:

void

Parameter:

[T_BOOLEAN](#) withUnit (in) *The flag if the attribute becomes an unit.*

Java Calling Sequence:

```
aaObj.withUnit(TRUE);
```

Since:

ASAM ODS 5.0

See also:

[hasUnit](#) of interface [ApplicationAttribute](#)

[setUnit](#) of interface [ApplicationAttribute](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

4.4.25 withValueFlag of interface ApplicationAttribute

Purpose:

Set whether the attribute becomes a value flag or not. If this flag isn't set the flag of the TS_Value will be ignored by the server.

Note:

It is only allowed to modify an attribute of an application element when there are no instance stored at the application element. Not yet implemented. Deprecated in ASAM ODS 5.2

Return:

void

Parameter:

[T_BOOLEAN](#) withValueFlag (in) *The flag if the attribute becomes a value flag.*

Java Calling Sequence:

```
aaObj.withValueFlag(TRUE);
```

Since:

ASAM ODS 5.0

See also:

[hasValueFlag](#) of interface [ApplicationAttribute](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)



4.5 ApplicationElement

Package:

org.asam.ods

Description:

The ASAM ODS application element interface.

4.5.1 createAttribute of interface ApplicationElement

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".

Note:

It is only allowed to create a new attribute, if there are no instances defined at the application element. For the transaction handling see section **Transaction handling in ODS API**, (p. 14)

Return:

[ApplicationAttribute](#) applAttr *The new application attribute.*

Parameter:

None.

Java Calling Sequence:

```
ApplicationAttribute applAttr = applElem.createAttribute();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_DUPLICATE_NAME](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testAddAttribute()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateElement()
test.highqsoft.odsapi.ApplModelCreateTest.testCreateAttr()
test.highqsoft.odsapi.ApplModelCreateTest.testSetBaseAttr()
test.highqsoft.odsapi.ApplModelCreateTest.testSetUnit()

```

4.5.2 createInstance of interface ApplicationElement**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.

Note:

See **Transaction handling in ODS API**.(p. 14)

Return:

[InstanceElement](#) instElem *The new instance.*

Parameter:

[Name](#) ieName (in) *The instance name.*

Java Calling Sequence:

```
InstanceElement instElem = applElem.createInstance(ieName);
```

Errors:

```

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
AO_TRANSACTION_NOT_ACTIVE

```

Tested by:

```

test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElem()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemAltUnit()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemModAttr()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemModLength()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemModObligatory()
test.highqsoft.odsapi.read.AsamPathTest.testCreateAsamPathSpaces()
test.highqsoft.odsapi.read.InstanceRelationTest.testCreateRecursiveInstances()
test.highqsoft.odsapi.read.ExtendedQueryTest.testCreateTestDataN_M()
test.highqsoft.odsapi.read.ExtendedQueryTest.testCreateTestDataOuterJoin()
test.highqsoft.odsapi.read.UnitTest.testSetValueInUnit()
....

```

4.5.3 createInstances of interface ApplicationElement

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.

Note:

See [Transaction handling in ODS API](#).(p. 14)

Return:

[InstanceElementSequence](#) elemList *The list with the new created instances.*

Parameter:

[NameValueSeqUnitSequence](#) attributes (in) *The attributes of the new created instances.*

[ApplicationRelationInstanceElementSeqSequence](#) relatedInstances (in) *The list with related instances for different application relations.*

Java Calling Sequence:

```
ApplicationElement aeObj;
NameValueSeqUnit attributes[];
ApplicationRelationInstanceElement[] relatedInstances;
InstanceElement elemList[] = aeObj.createInstances(attributes, relatedInstances);
```

Since:

ASAM ODS 5.0

Errors:

[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](#)

4.5.4 getAllRelatedElements of interface ApplicationElement

Purpose:

Get a list of all related application elements connected to this application element.

Return:

[ApplicationElementSequence](#) applElems *The related application elements.*

Parameter:

None.

Java Calling Sequence:

```
ApplicationElement[] applElems = applElem.getAllRelatedElements();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.odsapi.read.ExtendedQueryTest.testJoinSeqAttribute()
 test.highqsoft.odsapi.read.ExtendedQueryTest.testOuterJoin()

4.5.5 getAllRelations of interface ApplicationElement**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.

Return:

[ApplicationRelationSequence](#) applRels *The application relations of the application element.*

Parameter:

None.

Java Calling Sequence:

```
ApplicationRelation[] applRels = applElem.getAllRelations();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.ApplicationRelationTest.testApplicationInverseRelationRange()
 test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationNames()
 test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationType()
 test.highqsoft.avalon.InstanceElementTest.testInstancesRelationAppElemAccess()
 test.highqsoft.avalon.InstanceElementTest.testInstancesRelationNames()
 test.highqsoft.avalon.InstanceElementTest.testInstancesRelations()
 test.highqsoft.avalon.MultiSessionTest.threadInstancesRelations.run()
 test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationAppElemAccess.run()
 test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationNames.run()
 test.highqsoft.avalon.TableQueryTest.testTableQueryExt()

4.5.6 getApplicationStructure of interface ApplicationElement**Purpose:**

Get the application structure 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.

Return:

[ApplicationStructure](#) applStruct *The application structure to which the application element belongs.*

Parameter:

None.

Java Calling Sequence:

```
ApplicationStructure applStruct = applElem.getApplicationStructure();
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.5.7 getAttributeByBaseName of interface ApplicationElement

Purpose:

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.

Return:

[ApplicationAttribute](#) applAttr *The application attribute.*

Parameter:

[Name](#) baName (in) *The base attribute name.*

Java Calling Sequence:

```
ApplicationAttribute applAttr = applElem.getAttributeByBaseName(baName);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.ExtQueryTest.runCatalogElements()  
test.highqsoft.avalon.ExtQueryTest.runSimpleQueryExt()  
test.highqsoft.avalon.ExtQueryTest.testCatalogElements()  
test.highqsoft.avalon.ExtQueryTest.testCoSessCatalogElements()  
test.highqsoft.avalon.ExtQueryTest.testCoSessSimpleQueryExt()
```

```

test.highqsoft.avalon.ExtQueryTest.testSimpleQueryExt()
test.highqsoft.avalon.InstanceElementTest.testCheckRelLccMeq()
test.highqsoft.avalon.InstanceElementTest.testInstancesNamesQuery()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationApplElemAccess()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelations()
....

```

4.5.8 getAttributeByName of interface ApplicationElement

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 case sensitive, eg Id and ID are different application attributes, don't use this misleading attribute name.

Return:

[ApplicationAttribute](#) applAttr *The application attribute.*

Parameter:

[Name](#) aaName (in) *The application attribute name.*

Java Calling Sequence:

```
ApplicationAttribute applAttr = applElem.getAttributeByName(aaName);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadTestEnumeration-
Definition.run()
test.highqsoft.avalon.ApplicationStructureTest.testEnumerationDefinition()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCheckAttribute()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCheckDeleteAttribute()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCheckRename()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testDeleteAttribute()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testRenameAttribute()
test.highqsoft.odsapi.read.ExtendedQueryTest.testGroupByQueryExt()
test.highqsoft.odsapi.read.UnitTest.testGetInstancesExtAnuSeq()
test.highqsoft.odsapi.EnumerationTest.testReadEnumInstance()
....

```

4.5.9 getAttributes of interface ApplicationElement

Purpose:

Get a list of the application attributes of an application element. The reference attributes are not returned.



Return:

[ApplicationAttributeSequence](#) applAttrs *The application attributes.*

Parameter:

[Pattern](#) aaPattern (in) *The name or the search pattern for the application attribute name.*

Java Calling Sequence:

```
ApplicationAttribute[] applAttrs = applElem.getAttributes(aaPattern);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.SecurityTest.testPassword()
test.highqsoft.avalon.SecurityTest.testRights()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testIeCreateRelatedInstances()
test.highqsoft.odsapi.read.UnitTest.testGetInstancesAnuSeq()
test.highqsoft.odsapi.read.UnitTest.testGetInstancesCondSeq()
test.highqsoft.odsapi.read.ExtendedQueryTest.testMeqQtyQuery()
test.highqsoft.odsapi.read.UnitTest.testGetValueInUnit()
test.highqsoft.odsapi.security.AttributeCheckTest.testSetAttributeRights()
test.highqsoft.odsapi.security.InstanceCheckTest.testSubTestRights()
test.highqsoft.odsapi.ApplModelCreateTest.testCheckElement()
....
```

4.5.10 getBaseElement of interface ApplicationElement**Purpose:**

Get the base element of an application element.

Return:

[BaseElement](#) baseElem *The base element.*

Parameter:

None.

Java Calling Sequence:

```
BaseElement baseElem = applElem.getBaseElement();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationBase()
test.highqsoft.avalon.ApplicationStructureTest.testApplicationStructureValue()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testAddAttribute()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
test.highqsoft.odsapi.ApplModelCreateTest.testCheckElement()
test.highqsoft.odsapi.ApplModelCreateTest.testCreateElements()
test.highqsoft.odsapi.ApplModelCreateTest.testSetBaseAttr()

```

4.5.11 getId of interface ApplicationElement**Purpose:**

Get the Id of an application element.

Return:

T_LONGLONG aeId *The Id of the application element.*

Parameter:

None.

Java Calling Sequence:

```
T_LONGLONG aeId = applElem.getId();
```

Errors:

AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

```

test.highqsoft.avalon.ApplicationStructureTest.testApplicationStructureValue()
test.highqsoft.avalon.ExtQueryTest.runCatalogElements()
test.highqsoft.avalon.ExtQueryTest.runSimpleQueryExt()
test.highqsoft.avalon.ExtQueryTest.testCatalogElements()
test.highqsoft.avalon.ExtQueryTest.testCoSessCatalogElements()
test.highqsoft.avalon.ExtQueryTest.testCoSessSimpleQueryExt()
test.highqsoft.avalon.ExtQueryTest.testSimpleQueryExt()
test.highqsoft.avalon.InstanceElementTest.testCheckRelLccMeq()
test.highqsoft.avalon.InstanceElementTest.testInstancesAsamPath()
test.highqsoft.avalon.InstanceElementTest.testInstancesNamesQuery()
...

```

4.5.12 getInitialRightRelations of interface ApplicationElement**Purpose:**

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.



Return:

[ApplicationRelationSequence](#) applRels *The sequence with the application relations which will be used to create the initial rights of the new created instance element.*

Parameter:

None.

Java Calling Sequence:

```
ApplicationRelation[] applRels = applElem.getInitialRightRelations();
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.5.13 getInitialRights of interface ApplicationElement**Purpose:**

Retrieve access control list information for the initial rights of the given object.

Return:

[InitialRightSequence](#) initialRights *The access control list entries with the initial rights of the given application element.*

Parameter:

None.

Java Calling Sequence:

```
InitialRight[] initialRights = applElem.getInitialRights();
```

Java Example:

```
// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

// Get ApplElemAccess Interface
ApplElemAccess aea = aoSession.getApplElemAccess();

/* Get the all application elements. */
ApplicationElement aeList[] = as.getElements("");

for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {
    InitialRight [] irSeq;          // The Initial rights.

    // Name of the application element
    String aeName = aeList[aeIndex].getName();
    // Get the access control list
    irSeq = aeList[aeIndex].getRights();
    for (int irIndex = 0; irIndex < irSeq.length; irIndex++) {
        System.out.println(
            aeName +
            irSeq[irIndex].usergroupId.low + ", " +
            irSeq[irIndex].refAid.low + ", " +
            irSeq[irIndex].rights);
    }
}
```

See also:

[getElementInitialRightRelation](#) of interface [ApplElemAccess](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.SecurityTest.testIniRights()
 test.highqsoft.odsapi.XATFCopyTest.testCompareSecurityInformation()

4.5.14 getInstanceById of interface ApplicationElement

Purpose:

Get the instance element specified by the given Id. If the Id of the instance is not unique an exception is thrown.

Note:

This methods tries first to get the instance from the instancews cache, the instance cache will not be cleared independent from the INI-File variable NO_INSTANCE_CACHE.

Return:

[InstanceElement](#) instElem *The instance element.*

Parameter:

[T_LONGLONG](#) ieId (in) *The instance element Id.*

Java Calling Sequence:

```
InstanceElement instElem = applElem.getInstanceById(ieId);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.InstanceElementTest.testCheckRelLccMeq()
 test.highqsoft.avalon.InstanceElementTest.testCorbaReferences()
 test.highqsoft.avalon.InstanceElementTest.testInstancesNamesQuery()
 test.highqsoft.avalon.InstanceElementTest.testInstancesRelationApplElemAccess()
 test.highqsoft.avalon.MeasurementTest.testMeasurementQuery()
 test.highqsoft.avalon.MultiSessionTest.threadInstancesNamesQuery.run()
 test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationApplElemAccess.run()
 test.highqsoft.odsapi.read.AsamPathTest.pathTest()
 test.highqsoft.odsapi.read.ExtendedQueryTest.testAllSeqAttr()
 test.highqsoft.odsapi.read.InstanceRelationTest.testGetSmRelatedInstances()

4.5.15 getInstanceByName of interface ApplicationElement

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 convenience 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.

Return:

[InstanceElement](#) instElem *The instance element.*

Parameter:

[Name](#) ieName (in) *The instance element name.*

Java Calling Sequence:

```
InstanceElement instElem = applElem.getInstanceByName(ieName);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_DUPLICATE_NAME](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testAddAttribute()
test.highqsoft.odsapi.read.MeasurementTest.testValueMatrixUnit()
test.highqsoft.odsapi.read.UnitTest.testGetInstancesExtAnuSeq()
test.highqsoft.odsapi.read.UnitTest.testGetInstancesExtCondSeq()
test.highqsoft.odsapi.read.UnitTest.testSetValueInUnit()
test.highqsoft.odsapi.read.UnitTest.testValuematrixGet()
test.highqsoft.odsapi.security.AttributeCheckTest.testSetAttributeRights()
test.highqsoft.odsapi.security.InstanceCheckTest.testSubTestRights()
test.highqsoft.odsapi.security.LoginTest.testLogin()
test.highqsoft.odsapi.security.ManyGroupTest.testCreateGroups()
....
```

4.5.16 getInstances of interface ApplicationElement

Purpose:

Get the instances whose names match the pattern. The pattern is case sensitive and may contain wildcard characters.

Return:

[InstanceElementIterator](#) ieIterator *The instance elements.*

Parameter:

[Pattern](#) iePattern (in) *The name or the search pattern for the instance element name.*

Java Calling Sequence:

```
InstanceElementIterator ieIterator = applElem.getInstanceNames(iePattern);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.InstanceElementTest.testCorbaReferences()
test.highqsoft.avalon.InstanceElementTest.testInstanceRelatedAgainstInverseRelated()
test.highqsoft.avalon.InstanceElementTest.testInstanceNames()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationChild()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationNames()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelations()
test.highqsoft.avalon.MeasurementTest.testGetLocalColumnInstances()
test.highqsoft.avalon.MeasurementTest.testMeasurement()
test.highqsoft.avalon.MeasurementTest.testMeasurementCompareValues()
test.highqsoft.avalon.MeasurementTest.testMeasurementCompareValuesExt()
....

```

4.5.17 getName of interface ApplicationElement**Purpose:**

Get the name of an application element.

Return:

[Name](#) aeName *The name of the application element.*

Parameter:

None.

Java Calling Sequence:

```
Name aeName = applElem.getName();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.ApplicationRelationTest.testApplicationInverseRelationRange()
test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationNames()
test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationType()
test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadApplicationStructure-
Base.run()
test.highqsoft.avalon.ApplicationStructureTest.testApplicationStructureBase()

```



```

test.highqsoft.avalon.InstanceElementTest.testInstanceRelatedAgainstInverseRelated()
test.highqsoft.avalon.InstanceElementTest.testInstancesAsamPath()
test.highqsoft.avalon.InstanceElementTest.testInstancesNames()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationAppElemAccess()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationChild()
....

```

4.5.18 getRelatedElementsByRelationship of interface ApplicationElement

Purpose:

Get related application elements connected via the specified relationship.

Return:

[ApplicationElementSequence](#) applElems *The related application elements.*

Parameter:

[Relationship](#) aeRelationship (in) *The requested relationship.*

Java Calling Sequence:

```
ApplicationElement[] applElems = applElem.getRelatedElementsByRelationship(aeRelationship);
```

Errors:

```

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_INVALID_RELATIONSHIP
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

```

Tested by:

```

test.highqsoft.avalon.TableQueryTest.testTableQuery()
test.highqsoft.odsapi.CreateInstanceTestCase.createInstances()
test.highqsoft.odsapi.CreateInstanceTestCase.createMeasurementInstances()
test.highqsoft.odsapi.CreateMeasurementTest.testCreateMeasurementCommit()
test.highqsoft.odsapi.CreateMeasurementTest.testCreateMeasurementDoubleRel()
test.highqsoft.odsapi.DeleteInstanceElementTest.doubleDeleteInstances()
test.highqsoft.odsapi.UpdateMeasurementTest.createMeasurementBody()
test.highqsoft.odsapi.UpdateMeasurementTest.testUpdateMeasurement()

```

4.5.19 getRelationsByBaseName of interface ApplicationElement

Purpose:

Return the basic application relations derived from the base relation with the given relation name.

Take care at most elements only one application relation can be derived from a base relation, there are some well defined exceptions, so the method can return more then one application relation.

Return:

[ApplicationRelationSequence](#) applRel *The application relation sequence, if no relation is found an empty sequence is returned.*

Parameter:

[Name](#) baseRelName (in) *The name of the base relation.*

Java Calling Sequence:

```
ApplicationRelation applSeq[] = ae.getRelationaByBaseName("parent_test");
```

Since:

ASAM ODS 5.1.1

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationBase()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testIeCreateRelatedInstances()
test.highqsoft.odsapi.read.InstanceRelationTest.testCreateRecursiveInstances()
test.highqsoft.odsapi.read.InstanceRelationTest.testDeleteRecursiveInstances()
test.highqsoft.odsapi.read.InstanceRelationTest.testReadRecursiveInstances()
test.highqsoft.odsapi.read.ExtendedQueryTest.testGroupByQuery()
test.highqsoft.odsapi.WriteInstanceElementTest.testDoubleRelations()
test.highqsoft.odsapi.WriteInstanceElementTest.testWriteMultiLcs()
```

4.5.20 getRelationsByType of interface ApplicationElement

Purpose:

Get application relations of the requested type connected from this application element. The inverse relations are not returned.

Return:

[ApplicationRelationSequence](#) applRels *The application relations.*

Parameter:

[RelationType](#) aeRelationType (in) *The requested relation type.*

Java Calling Sequence:

```
ApplicationRelation[] applRels = applElem.getRelationsByType(aeRelationType);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_RELATION_TYPE](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)



Tested by:

test.highqsoft.odsapi.DeleteInstanceElementTest.deleteInstances()

4.5.21 getRights of interface ApplicationElement**Purpose:**

Retrieve access control list information of the given object.

Return:

[ACLSequence](#) aclEntries *The access control list entries of the given application element.*

Parameter:

None.

Java Calling Sequence:

```
ACL[] aclEntries = applElem.getRights();
```

Java Example:

```
// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

// Get ApplElemAccess Interface
ApplElemAccess aea = aoSession.getApplElemAccess();

/* Get the all application elements. */
ApplicationElement aeList[] = as.getElements("");

for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {
    ACL [] acList;           // The access control list.

    // Name of the application element
    String aeName = aeList[aeIndex].getName();
    // Get the access control list
    acList = aeList[aeIndex].getRights();
    for (int aclIndex = 0; aclIndex < acList.length; aclIndex++) {
        System.out.println(
            aeName + ", 0, " +
            acList[aclIndex].userId + " ", " +
            acList[aclIndex].rights);
    }
}
```

See also:

[getElementRights](#) of interface [ApplElemAccess](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.SecurityTest.testRights()

4.5.22 getSecurityLevel of interface ApplicationElement

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.

Return:

[T_LONG](#) secLevel *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.*

Parameter:

None.

Java Calling Sequence:

```
SecurityLevel secLevel = applElem.getSecurityLevel();
```

Java Example:

```
// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

if (as != null) {

    // Get all application elements.
    ApplicationElement aeList[] = as.getElements("*");

    for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {

        int secLevel;                // securtiy level

        secLevel = aeList[aeIndex].getSecurityLevel();
        System.out.println(aeList[aeIndex].getName() + ", " +
            secLevel);
    }
}
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.5.23 listAllRelatedElements of interface ApplicationElement

Purpose:

Get the names of all related application elements.

Return:

[NameSequence](#) applElemNames *The names of the related application elements.*

Parameter:

None.



Java Calling Sequence:

```
Name[] applElemNames = applElem.listAllRelatedElements();
```

Errors:

AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

4.5.24 listAttributes of interface ApplicationElement**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.

Return:

[NameSequence](#) applAttrNames *The names of the application attributes.*

Parameter:

[Pattern](#) aaPattern (in) *The name or the search pattern for the application attribute name.*

Java Calling Sequence:

```
Name[] applAttrNames = applElem.listAttributes(aaPattern);
```

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

```
test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadApplicationElement.run()  
test.highqsoft.avalon.ApplicationStructureTest.testApplicationElement()  
test.highqsoft.odsapi.InstanceAttributeTest.testRenameInstanceAttribute()  
test.highqsoft.odsapi.InstanceAttributeTest.testRemoveInstanceAttribute()
```

4.5.25 listInstances of interface ApplicationElement**Purpose:**

Get the names of the instances whose names match the pattern. The pattern is case sensitive and may contain wildcard characters.

Return:

[NameIterator](#) ieNameIterator *The names of the instances.*

Parameter:

[Pattern](#) aaPattern (in) *The name or the search pattern for the application attribute name.*

Java Calling Sequence:

```
NameIterator ieNameIterator = applElem.listInstances(aaPattern);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.InstanceElementTest.testInstancesNames()
test.highqsoft.odsapi.read.UnitTest.testGetInstancesExtAnuSeq()
test.highqsoft.odsapi.DeleteInstanceElementTest.testApplElemAccessDeleteInstances()
```

4.5.26 listRelatedElementsByRelationship of interface Application-Element

Purpose:

Get the names of related application elements connected via the specified relationship.

Return:

[NameSequence](#) applElemNames *The names of the related application elements.*

Parameter:

[Relationship](#) aeRelationship (in) *The requested relationship.*

Java Calling Sequence:

```
Name[] applElemNames = applElem.listRelatedElementsByRelationship(aeRelationship);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_RELATIONSHIP](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.5.27 removeAttribute of interface ApplicationElement

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.

Note:

See **Transaction handling in ODS API.**(p. 14)



Return:

void

Parameter:[ApplicationAttribute](#) applAttr (in) *The application attribute to remove.***Java Calling Sequence:**`applElem.removeAttribute(applAttr);`**Errors:**

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_FOUND](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:`test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testDeleteAttribute()`**4.5.28 removeInstance of interface ApplicationElement****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.

Note:See **Transaction handling in ODS API**.(p. 14)**Return:**

void

Parameter:[T_LONGLONG](#) ieId (in) *The instance Id.*[T_BOOLEAN](#) recursive (in) *The recursive flag.***Java Calling Sequence:**`applElem.removeInstance(ieId,recursive);`**Errors:**

[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
 AO_TRANSACTION_NOT_ACTIVE

Tested by:

```

test.highqsoft.odsapi.security.AttributeCheckTest.testSetAttributeRights()
test.highqsoft.odsapi.CreateInstanceElementTest.XXtestDeleteMeaInstances()
test.highqsoft.odsapi.UpdateInstanceElementTest.testDeleteMeasurementQuantity()
test.highqsoft.odsapi.UpdateInstanceElementTest.testReplaceLocalColumn()
test.highqsoft.odsapi.DeleteInstanceElementTest.deleteInstances()
test.highqsoft.odsapi.DeleteInstanceElementTest.testDIADemDelete()
test.highqsoft.odsapi.InstanceAttributeTest.testDeleteInstanceAttribute()
test.highqsoft.odsapi.UpdateMeasurementTest.testDeleteMeasurement()
test.highqsoft.odsapi.WriteInstanceElementTest.testDuplicateNameAEA()
test.highqsoft.odsapi.WriteInstanceElementTest.testLongNameAEA()
....
  
```

4.5.29 setBaseElement of interface ApplicationElement

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.

Note:

See **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

[BaseElement](#) baseElem (in) *The base element.*

Java Calling Sequence:

```
applElem.setBaseElement(baseElem);
```

Errors:

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
 AO_TRANSACTION_NOT_ACTIVE



4.5.30 setInitialRightRelation of interface ApplicationElement

Purpose:

Set for the given application element, which relation will be used to determine the initial rights for the new created instances.

Note:

See [Transaction handling in ODS API](#).(p. 14)

Return:

void

Parameter:

[ApplicationRelation](#) applRel (in) *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 an unique instance element to retrieve the initial rights.*

[T_BOOLEAN](#) set (in) *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);
```

See also:

[setElementInitialRightRelation](#) of interface [ApplElemAccess](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

4.5.31 setInitialRights of interface ApplicationElement

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.

Return:

void

Parameter:

[InstanceElement](#) usergroup (in) *The usergroup for which the initial rights will be modified.*

[T_LONG](#) rights (in) *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.*

T_LONGLONG refAid (in) *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.*

RightsSet set (in) *What to do with the new initial rights.*

Java Calling Sequence:

```
applElem.setInitialRights(usergroup, rights, refAid, set);
```

See also:

[setElementInitialRights](#) of interface [ApplElemAccess](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.security.ApplModelCheckTest.testSetIniRights()  
test.highqsoft.odsapi.security.SecurityTestCase.setIniRights()
```

4.5.32 setName of interface ApplicationElement

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. Current server implementations restrict it to 30 characters.

Note:

See **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

Name aeName (in) *The application element name.*

Java Calling Sequence:

```
applElem.setName(aeName);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_DUPLICATE_NAME](#)



[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
 test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
 test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateElement()
 test.highqsoft.odsapi.ApplModelCreateTest.testCreateElements()
 test.highqsoft.odsapi.EnumerationTest.testWriteApplicationModel()

4.5.33 setRights of interface ApplicationElement**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.

Note:

See **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

[InstanceElement](#) usergroup (in) *The usergroup for which the rights will be modified.*

[T_LONG](#) rights (in) *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.*

[RightsSet](#) set (in) *What to do with the new right.*

Java Calling Sequence:

```
applElem.setRights(usergroup, rights, set);
```

See also:

[setElementRights](#) of interface [ApplElemAccess](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.odsapi.security.ApplModelCheckTest.testElementSecurity()
test.highqsoft.odsapi.security.ApplModelCheckTest.testSecurityLevel()
test.highqsoft.odsapi.security.ApplModelCheckTest.testSetIniRights()
test.highqsoft.odsapi.security.ApplModelCheckTest.testSetRights()
test.highqsoft.odsapi.security.AttributeCheckTest.testSetAttributeRights()
test.highqsoft.odsapi.security.InstanceCheckTest.testSetInstanceInitialInstRights()
test.highqsoft.odsapi.security.InstanceCheckTest.testSetInstanceInitialRights()
test.highqsoft.odsapi.security.InstanceCheckTest.testSetInstanceInstRights()
test.highqsoft.odsapi.security.SecurityTest Case.setRights()
test.highqsoft.odsapi.CreateInstanceElementTest.testCreateElement Rights()

```

4.5.34 setSecurityLevel of interface ApplicationElement**Purpose:**

Set the security level for the given application element. If the security level is added the client is responsible for the access control list entries of the existing objects.

Note:

See **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

T_LONG secLevel (in) *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.*

RightsSet set (in) *What to do with the new security level.*

Java Calling Sequence:

```
applElem.setSecurityLevel(secLevel,set);
```

Errors:

```

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
AO_TRANSACTION_NOT_ACTIVE

```

Tested by:

```

test.highqsoft.odsapi.security.ApplModelCheckTest.testElementSecurity()
test.highqsoft.odsapi.security.ApplModelCheckTest.testSecurityLevel()
test.highqsoft.odsapi.security.AttributeCheckTest.testSetAttributeRights()
test.highqsoft.odsapi.security.InstanceCheckTest.testCurrentRights()
test.highqsoft.odsapi.security.InstanceCheckTest.testSetInstanceInitialInstRights()
test.highqsoft.odsapi.security.InstanceCheckTest.testSetInstanceInstRights()
test.highqsoft.odsapi.security.InstanceCheckTest.testSubTest Rights()
test.highqsoft.odsapi.security.SecurityTest Case.setSecurityLevel()
test.highqsoft.odsapi.CreateInstanceElementTest.testCreateElement Rights()

```



4.6 ApplicationRelation

Package:

org.asam.ods

Description:

The ASAM ODS application relation interface.

A relation is the connection between two ASAM-ODS elements in both directions. There is no inverse relation, the Upper-API allows to ask 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 works from the second element.

E.g.

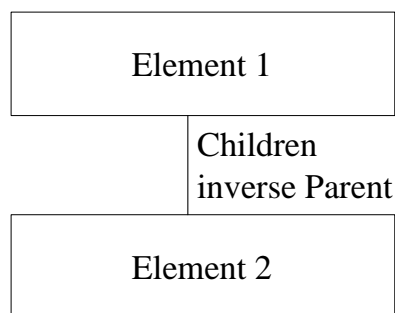


Figure 4.1: Example of relation

method `getRelationName` returns "Children" method `getInverseRelationName` returns "Parent"

method `getElem1` will return 'Element 1' method `getElem2` will return 'Element 2'

When we create a relation between two elements the inverse way is also created and when we remove a relation the inverse way is also removed. This is valid for the application structure and the instance structure.

When a new relation is created take care the correct 'Element 1' and 'Element 2' are given at the corresponding set function.

4.6.1 The Relation Type

The [RelationType](#) is an ASAM ODS defined enumeration. There are three types of the realtions:

FATHER_ CHILD

there is a hierachical relation between two elements. The FATHER_ CHILD relation is used for the uniqueness of the instance by the ASAM-PATH.

E.g. AoMeasurement and AoMeasurementQuantity.

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 basetype.

E.g. AoSubTest and AoSubTest.

4.6.2 The Relationship

The [RelationShip](#) is an ASAM ODS defined enumeration. Beside 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 doesn't 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 which application element of the instance elements are the 'element 1' or 'element 2' of the application relation.

4.6.3 The Relation Range

The [RelationRange](#) is an ASAM ODS defined structure. The relation range gives the number of the allowed relation 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 -1 Many).

For the relation range the methods `getRelationRange`, `getInverseRelationRange`, `setRelationRange` and `setInverseRelationRange` are introduced.

Using this relation range the obligatory flag is not longer needed at the relation, so the corresponding methods on the interfaces `BaseRelation` and `ApplicationRelation` are also not longer needed. If the minimum number of the relation range is 1 then the relation is required.

If there is a base reference the relation type and the relation range of the base reference are used at the application reference. If there is no base reference the default relation type will be `INFO` and a relationship `INFO_TO`. Depending on the datatype of the attribute the relation range is set. A datatype of `DT_*` means a relation range of 0:1. A datatype of `DS_*` means a relation range of 0:MANY.

The relation range has the following structure:

`typedef struct RELATIONRANGE RelationRange;`

```
struct RELATIONRANGE {
    T_SHORT    min;
    T_SHORT    max;
};
```

The fields means:

min

The minimum number of relations, 0 or 1. If the minimum value is 1 there must be always a reference set. The value -2 means not initialised.

max

The maximum number of relation. The value -1 means MANY. There is no specified maximum any amount greater equal minimum is allowed. The value -2 means not initialised.

E.g

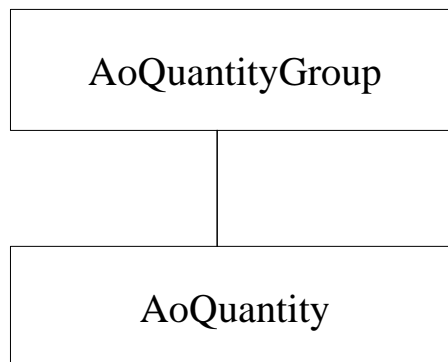


Figure 4.2: Example of N:M relation

4.6.4 Modification of the relation type.

It is only allowed to modify a relation type when no base relation is given, otherwise the exception `AO_HAS_BASE_RELATION` is thrown.

It is not allowed to set the relation type `FATHER_CHILD`, these relation type is reserved for the base model. The two relationtypes which can be set are `INFO` or `INHERITANCE`.

The relationship is set automatic when the relation type is set, the maximum of the relation range determines the relationship between the elements of the relation.

RelationType	maximum RelationRange	maximum inverse RelationRange	Relationship
INFO	1	1	exception, <code>AO_INVALID_- RELATION_TYPE</code>
INFO	1	MANY	<code>INFO_TO</code>
INFO	MANY	1	<code>INFO_FROM</code>
INFO	MANY	MANY	<code>INFO</code>
INHERITANCE	1	1	exception, <code>AO_INVALID_- RELATION_TYPE</code>
INHERITANCE	1	MANY	<code>SUPERTYPE</code>
INHERITANCE	MANY	1	<code>SUBTYPE</code>
INHERITANCE	MANY	MANY	exception, <code>AO_INVALID_- RELATION_TYPE</code>

The method `setRelationType` of interface `ApplicationRelation`(p. 154) allows modification of the relationtype.

4.6.5 `getBaseRelation` of interface `ApplicationRelation`

Purpose:

Get the base relation of the application relation.

Return:

[BaseRelation](#) baseRel *The base relation of the application relation. A 'null' is returned if the application relation has no base relation.*

Parameter:

None.

Java Calling Sequence:

```
BaseRelation baseRel = applRel.getBaseRelation();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.6.6 getElem1 of interface ApplicationRelation

Purpose:

Get the first application element of the application relation.

Return:

[ApplicationElement](#) applElem *The first application element of the application relation.*

Parameter:

None.

Java Calling Sequence:

```
ApplicationElement applElem = applRel.getElem1();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.TableQueryTest.testTableQueryExt()  
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testIeCreateRelatedInstances()  
test.highqsoft.odsapi.DeleteInstanceElementTest.deleteInstances()
```

4.6.7 getElem2 of interface ApplicationRelation

Purpose:

Get the second application element of the application relation.

Return:

[ApplicationElement](#) applElem *The second application element of the application relation.*

Parameter:

None.

Java Calling Sequence:

```
ApplicationElement applElem = applRel.getElem2();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.TableQueryTest.testTableQueryExt()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testIeCreateRelatedInstances()
test.highqsoft.odsapi.DeleteInstanceElementTest.deleteInstances()
test.highqsoft.odsapi.UpdateInstanceElementTest.testUpdateExtRef()
test.highqsoft.odsapi.WriteInstanceElementTest.testDoubleRelations()
test.highqsoft.odsapi.WriteInstanceElementTest.testWriteMultiLcs()

```

4.6.8 getInverseRelationName of interface ApplicationRelation**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.

Return:

[Name](#) arInvName *The inverse name of the application relation.*

Parameter:

None.

Java Calling Sequence:

```
Name arInvName = applRel.getInverseRelationName();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.ApplicationRelationTest.testApplicationInverseRelationRange()
test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationNames()
test.highqsoft.odsapi.read.QueryTest.testRelName()
test.highqsoft.odsapi.security.InstanceCheckTest.testSubTestRights()

```


4.6.9 getInverseRelationRange of interface ApplicationRelation

Purpose:

Get the inverse relation range of the application relation.

Return:

[RelationRange](#) arRange *The inverse relation range of the application relation.*

Parameter:

None.

Java Calling Sequence:

```
RelationRange arRange = applRel.getInverseRelationRange();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.ApplicationRelationTest.testApplicationInverseRelationRange()
```

4.6.10 getInverseRelationship of interface ApplicationRelation

Purpose:

Get the inverse relationship of the application relation.

Return:

[Relationship](#) relationship *The inverse relationship of the application relation.*

Parameter:

None.

Java Calling Sequence:

```
Relationship relationship = applRel.getInverseRelationship();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.6.11 getRelationName of interface ApplicationRelation

Purpose:

Get the name of the application relation.

Return:

[Name](#) arName *The name of the application relation.*

Parameter:

None.

Java Calling Sequence:

```
Name arName = applRel.getRelationName();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationBase()
test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationNames()
test.highqsoft.avalon.ExtQueryTest.runCatalogElements()
test.highqsoft.avalon.ExtQueryTest.testCatalogElements()
test.highqsoft.avalon.ExtQueryTest.testCoSessCatalogElements()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationApplElemAccess()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationNames()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelations()
test.highqsoft.avalon.MeasurementTest.testMeasurementQuery()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelations.run()
....
```

4.6.12 getRelationRange of interface ApplicationRelation

Purpose:

Get the relation range of the application relation.

Return:

[RelationRange](#) arRange *The relation range of the application relation.*

Parameter:

None.

Java Calling Sequence:

```
RelationRange arRange = applRel.getRelationRange();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.TableQueryTest.getRelAttributeName()
test.highqsoft.odsapi.security.InstanceCheckTest.testSubTestRights()
test.highqsoft.odsapi.WriteInstanceElementTest.testInsertInstancesWithRights()
test.highqsoft.odsapi.WriteInstanceElementTest.testReadInstancesWithRights()
```



4.6.13 getRelationship of interface ApplicationRelation

Purpose:

Get the relationship of the application relation.

Return:

[Relationship](#) relationship *The relationship of the application relation.*

Parameter:

None.

Java Calling Sequence:

```
Relationship relationship = applRel.getRelationship();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.6.14 getRelationType of interface ApplicationRelation

Purpose:

Get the relation type of the application relation.

Return:

[RelationType](#) arType *The relation type of the application relation.*

Parameter:

None.

Java Calling Sequence:

```
RelationType arType = applRel.getRelationType();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.6.15 setBaseRelation of interface ApplicationRelation

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.

Note:

It is only allowed to modify the application relation if there are no instances at both application elements. See also **Transaction handling in ODS API.**(p. 14)

Return:

void

Parameter:

[BaseRelation](#) baseRel (in) *The base relation.*

Java Calling Sequence:

```
applRel.setBaseRelation(baseRel);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_RELATION](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()  
test.highqsoft.odsapi.ApplModelCreateTest.testCreateElements()
```

4.6.16 setElem1 of interface ApplicationRelation

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.

Note:

It is only allowed to modify the application relation if there are no instances at both application elements. See also **Transaction handling in ODS API.**(p. 14)

Return:

void

Parameter:

[ApplicationElement](#) applElem (in) *The application element.*

Java Calling Sequence:

```
applRel.setElem1(applElem);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_ELEMENT](#)



[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
test.highqsoft.odsapi.ApplModelCreateTest.testCreateElements()

```

4.6.17 setElem2 of interface ApplicationRelation

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.

Note:

It is only allowed to modify the application relation if there are no instances at both application elements. See also **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

[ApplicationElement](#) applElem (in) *The application element.*

Java Calling Sequence:

```
applRel.setElem2(applElem);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_ELEMENT](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
test.highqsoft.odsapi.ApplModelCreateTest.testCreateElements()

```

4.6.18 setInverseRelationName of interface ApplicationRelation

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.

Note:

It is only allowed to modify the application relation if there are no instances at both application elements. See also **Transaction handling in ODS API.**(p. 14)

Return:

void

Parameter:

[Name](#) arInvName (in) *The inverse application relation name.*

Java Calling Sequence:

```
applRel.setInverseRelationName(arInvName);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
test.highqsoft.odsapi.ApplModelCreateTest.testCreateElements()
```

4.6.19 setInverseRelationRange of interface ApplicationRelation

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.

Note:

It is only allowed to modify the application relation if there are no instances at both application elements. See also **Transaction handling in ODS API.**(p. 14)

Return:

void

Parameter:

[RelationRange](#) arRelationRange (in) *The inverse relation range.*

Java Calling Sequence:

```
applRel.setInverseRelationRange(arRelationRange);
```

Errors:

[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

Tested by:

test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()

4.6.20 setRelationName of interface ApplicationRelation

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 attribute must not exceed the maximum name length of the underlying physical storage. Current server typically implementations restrict it to 30 characters.

Note:

It is only allowed to modify the application relation if there are no instances at both application elements. See also **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

Name arName (in) *The application relation name.*

Java Calling Sequence:

```
applRel.setRelationName(arName);
```

Errors:

AO_BAD_PARAMETER
 AO_CONNECTION_LOST
 AO_IMPLEMENTATION_PROBLEM
 AO_NOT_IMPLEMENTED
 AO_NO_MEMORY
 AO_SESSION_NOT_ACTIVE

Tested by:

test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
 test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
 test.highqsoft.odsapi.ApplModelCreateTest.testCreateElements()

4.6.21 setRelationRange of interface ApplicationRelation

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.

Note:

It is only allowed to modify the application relation if there are no instances at both application elements. See also **Transaction handling in ODS API.**(p. 14)

Return:

void

Parameter:

[RelationRange](#) arRelationRange (in) *The relation range.*

Java Calling Sequence:

```
applRel.setRelationRange(arRelationRange);
```

Errors:

[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](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
```

4.6.22 setRelationType of interface ApplicationRelation

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.

It is only allowed to modify a relation type when no base relation is given, otherwise the exception [AO_HAS_BASE_RELATION](#) is thrown.

It is not allowed to set the relation type [FATHER_CHILD](#), these relation type is reserved for the base model. The two relationtypes which can be set are [INFO](#) or [INHERITANCE](#).

The relationship is set automatic when the relation type is set, the maximum of the relation range determines the relationship between the elements of the relation.

R.Type | max R.Range | max inv. R.Range | Relationship

INFO	1	1	exception, AO_INVALID_RELATION_TYPE
------	---	---	---

INFO	1	MANY	INFO_TO
------	---	------	-------------------------

INFO	MANY	1	INFO_FROM
------	------	---	---------------------------

INFO	MANY	MANY	INFO
------	------	------	----------------------



INHERITANCE	1	1	exception, AO_INVALID_RELATION_TYPE
INHERITANCE	1	MANY	SUPERTYPE
INHERITANCE	MANY	1	SUBTYPE
INHERITANCE	MANY	MANY	exception, AO_INVALID_RELATION_TYPE

Note:

It is only allowed to modify the application relation if there are no instances at both application elements. See also **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

[RelationType](#) arRelationType (in) *The relation type.*

Java Calling Sequence:

```
applRel.setRelationType(arRelationType);
```

Errors:

[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](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

4.7 ApplicationStructure

Package:

org.asam.ods

Description:

The ASAM ODS application structure interface.

4.7.1 check of interface ApplicationStructure

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 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.

Note:

This method is not complete implemented.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
applStruct.check();
```

Errors:

[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](#)

Tested by:

```
org.asam.ods.ApplicationStructurePOA._invoke()
test.highqsoft.avalon.ApplicationStructureTest.testApplicationStructureCheck()
```

4.7.2 createElement of interface ApplicationStructure

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.

Note:

It is only allowed to modify the application structure if the involved application elements have no instances. See also **Transaction handling in ODS API**.(p. 14)

Return:

[ApplicationElement](#) applElem *The new application element.*



Parameter:

[BaseElement](#) baseElem (in) *The base element from which the application element is derived.*

Java Calling Sequence:

```
ApplicationElement applElem = applStruct.createElement(baseElem);
```

Java Example:

```
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 structure.
    ApplicationStructure as = session.getApplicationStructure();
    // Get base structure.
    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();
}
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateElement()
test.highqsoft.odsapi.ApplModelCreateTest.testCreateElements()
test.highqsoft.odsapi.EnumerationTest.testWriteApplicationModel()
```

4.7.3 createEnumerationDefinition of interface ApplicationStructure**Purpose:**

Create a new enumeration definition. This method modifies the application model and is only allowed for the superuser.

Note:

See also **Transaction handling in ODS API.**(p. 14)

Return:

[EnumerationDefinition](#) newEnum *The new created enumeration*

Parameter:

[T_STRING](#) enumName (in) *Name of the enumeration*

Java Calling Sequence:

```
EnumerationDefinition enumDef asObj.createEnumerationDefinition(enumName);
```

Since:

ASAM ODS 5.0

Errors:

[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](#)

Tested by:

```
test.highqsoft.odsapi.EnumerationTest.testWriteApplicationModel()
```

4.7.4 createInstanceRelations of interface ApplicationStructure

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.

Note:

See also **Transaction handling in ODS API.**(p. 14)

Return:

void

Parameter:

[ApplicationRelation](#) applRel (in) *The application relation.*

[InstanceElementSequence](#) elemList1 (in) *The list with the instances of one application element for which the relation will be created.*

[InstanceElementSequence](#) elemList2 (in) *The list with the related instances.*

Java Calling Sequence:

```
ApplicationStructure as;  
InstanceElement elemList1[];  
InstanceElement elemList2[];  
ApplicationRelation applRel;  
as.createInstanceRelations(applRel, elemList1, elemList2);
```

Since:

ASAM ODS 5.0

Errors:

[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](#)

Tested by:

test.highqsoft.odsapi.WriteInstanceElementTest.testCreateInstanceRelations()

4.7.5 createRelation of interface ApplicationStructure**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".

Note:It is only allowed to modify the application structure if the involved application elements have no instances. See also **Transaction handling in ODS API.**(p. 14)**Return:**[ApplicationRelation](#) applRel *The new application relation.***Parameter:**

None.

Java Calling Sequence:

```
ApplicationRelation applRel = applStruct.createRelation();
```

Java Example:

```

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 structure.
    ApplicationStructure as = session.getApplicationStructure();

    // Create a new application relation.
    ApplicationRelation ar = as.createRelation();

    ...
    session.commitTransaction();
    // Close the session.
    session.close();
}

```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
test.highqsoft.odsapi.ApplModelCreateTest.testCreateElements()

```

4.7.6 getElementById of interface ApplicationStructure

Purpose:

Get the application element with the requested Id.

Return:

[ApplicationElement](#) applElem *The requested application element.*

Parameter:

[T_LONGLONG](#) aeId (in) *The Id of the requested application element.*

Java Calling Sequence:

```
ApplicationElement applElem = applStruct.getElementById(aeId);
```

Java Example:

```

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 structure.
ApplicationStructure as = session.getApplicationStructure();

// Get an application element by id.
ApplicationElement ae = as.getElementById(69);

...

// Close the session.
session.close();
}

```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationStructure()
test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadTestEnumeration-
Definition.run()
test.highqsoft.avalon.ApplicationStructureTest.testEnumerationDefinition()

```

4.7.7 getElementByName of interface ApplicationStructure**Purpose:**

Get the application element with the requested name.

Return:

[ApplicationElement](#) applElem *The requested application element.*

Parameter:

Name aeName (in) *The name of the requested application element.*

Java Calling Sequence:

```
ApplicationElement applElem = applStruct.getElementByName(aeName);
```

Java Example:

```

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 structure.
    ApplicationStructure as = session.getApplicationStructure();
}

```

```

// Get an application element by name.
ApplicationElement ae = as.getElementByName("myEngine");

...

// Close the session.
session.close();
}

```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.TableQueryTest.testTableQuery()
test.highqsoft.avalon.TableQueryTest.testTableQueryExt()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testAddAttribute()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCheckAttribute()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCheckDeleteAttribute()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCheckRename()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElem()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemAltUnit()
....

```

4.7.8 getElements of interface ApplicationStructure

Purpose:

Get the application elements whose names match the pattern. The pattern is case sensitive and may contain wildcard characters.

Return:

[ApplicationElementSequence](#) applElems *The requested application elements.*

Parameter:

[Pattern](#) aePattern (in) *The name or the search pattern for the requested application elements.*

Java Calling Sequence:

```
ApplicationElement[] applElems = applStruct.getElements(aePattern);
```

Java Example:

```

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 structure.
    ApplicationStructure as = session.getApplicationStructure();

    // Get a list of application elements.
    ApplicationElementSeq aeSeq[] = as.getElements("*");

    ...

    // Close the session.
    session.close();

}
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.ApplicationRelationTest.testApplicationInverseRelationRange()
test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationBase()
test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationNames()
test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationType()
test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadApplicationElement.run()
test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadApplicationStructure-
Base.run()
test.highqsoft.avalon.ApplicationStructureTest.testApplicationElement()
test.highqsoft.avalon.ApplicationStructureTest.testApplicationElementUnique()
test.highqsoft.avalon.ApplicationStructureTest.testApplicationStructureBase()
test.highqsoft.avalon.ApplicationStructureTest.testApplicationStructureValue()
....
```

4.7.9 getElementsByBaseType of interface ApplicationStructure**Purpose:**

Get the names of application elements that are derived from the specified base element.

Return:

[ApplicationElementSequence](#) applElems *The requested application element names.*

Parameter:

[BaseType](#) aeType (in) *The requested base element type. The base element type can be a pattern.*

Java Calling Sequence:

```
ApplicationElement[] applElems = applStruct.getElementsByBaseType(aeType);
```

Java Example:

```
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 structure.
    ApplicationStructure as = session.getApplicationStructure();

    // Get a list of application elements by type.
    ApplicationElementSeq aeSeq[] = as.getElementsByBaseType("AoTest");

    ...

    // Close the session.
    session.close();

}

```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_BASETYPE](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.ApplicationStructureTest.testGetInstances()
test.highqsoft.avalon.MeasurementTest.testManyMeasurement()
test.highqsoft.avalon.MeasurementTest.testMeasurement()
test.highqsoft.avalon.MeasurementTest.testMeasurementQuery()
test.highqsoft.avalon.MultiSessionTest.threadCreateValueMatrix.run()
test.highqsoft.avalon.RelationTest.testRelationNames()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrix()
test.highqsoft.odsapi.changemodel.CreateNewInstanceElementTest.testCreateNew-
Instances()
test.highqsoft.odsapi.read.AsamPathTest.pathTest()
test.highqsoft.odsapi.read.AsamPathTest.testCreateAsamPathSpaces()
....

4.7.10 getEnumerationDefinition of interface ApplicationStructure**Purpose:**

Get the specified enumeration definition.

Return:

[EnumerationDefinition](#) enumDef *The enumeration definition.*

Parameter:

[T_STRING](#) enumName (in) *Name of the requested enumeration.*



Java Calling Sequence:

```
EnumerationDefinition enumDef = asObj.getEnumerationDefinition(enumName);
```

Since:

ASAM ODS 5.0

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.EnumerationTest.testCreateEnumerationDefinition()
```

4.7.11 getInstanceByAsamPath of interface ApplicationStructure

Purpose:

Get the instance element specified by the ASAM path.

Return:

[InstanceElement](#) instElem *The requested instance element.*

Parameter:

[Name](#) asamPath (in) *The ASAM path of the requested instance element.*

Java Calling Sequence:

```
InstanceElement instElem = applStruct.getInstanceByAsamPath(asamPath);
```

Java Example:

```
InstanceElement ie;  
Name asamPath;  
ie = as.getInstanceByAsamPath(asamPath);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_ASAM_PATH](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.InstanceElementTest.testInstancesAsamPath()  
test.highqsoft.avalon.MultiSessionTest.threadInstancesAsamPath.run()  
test.highqsoft.odsapi.read.AsamPathTest.testCheckAsamPathSpaces()  
test.highqsoft.odsapi.UpdateInstanceElementTest.updateGlobalFlag()
```

4.7.12 `getInstancesById` of interface `ApplicationStructure`

Purpose:

Get the instance elements specified by the element id.

Return:

[InstanceElementSequence](#) instElems *The requested instance element sequence.*

Parameter:

[ElemIdSequence](#) ieIds (in) *The sequence with the element id.*

Java Calling Sequence:

```
InstanceElement[] instElems = applStruct.getInstancesById(ieIds);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_ASAM_PATH](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.InstanceElementTest.testInstancesAsamPath()
test.highqsoft.avalon.MultiSessionTest.threadInstancesAsamPath.run()
test.highqsoft.odsapi.read.UnitTest.testValuematrixCheck()
test.highqsoft.odsapi.DeleteInstanceElementTest.deleteInstances()
```

4.7.13 `getRelations` of interface `ApplicationStructure`

Purpose:

Returns the relations between two application elements.

Return:

[ApplicationRelationSequence](#) applRels *The relations between the specified application elements.*

Parameter:

[ApplicationElement](#) applElem1 (in) *The first application element.*

[ApplicationElement](#) applElem2 (in) *The second application element.*

Java Calling Sequence:

```
ApplicationRelation[] applRels = applStruct.getRelations(applElem1, applElem2);
```

Java Example:

```
ApplicationRelationSeq arSeq[];
ApplicationElement applElem1, applElem2;
arSeq = as.getRelations(applElem1, applElem2);
```



Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationStructure()
 test.highqsoft.avalon.InstanceElementTest.testInstanceRelatedAgainstInverseRelated()
 test.highqsoft.avalon.MeasurementTest.testMeasurement()
 test.highqsoft.avalon.MeasurementTest.testMeasurementQuery()
 test.highqsoft.avalon.RelationTest.testRelationNames()
 test.highqsoft.odsapi.read.QueryTest.testRelName()
 test.highqsoft.odsapi.read.ExtendedQueryTest.testMultiElements()
 test.highqsoft.odsapi.ApplModelCreateTest.testCorbaException()

4.7.14 getSession of interface ApplicationStructure**Purpose:**

Get the current client session in which the application model is created.

Return:

[AoSession](#) session *The current client session.*

Parameter:

None.

Java Calling Sequence:

```
AoSession session = applStruct.getSession();
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.7.15 getTopLevelElements of interface ApplicationStructure**Purpose:**

Get the top level application elements which are inherited 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.

Return:

[ApplicationElementSequence](#) applElems *The top level application elements.*

Parameter:

[BaseType](#) aeType (in) *The requested base type. The base element type can be a pattern.*

Java Calling Sequence:

```
ApplicationElement[] applElems = applStruct.getTopLevelElements(aeType);
```

Java Example:

```
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 structure.
    ApplicationStructure as = session.getApplicationStructure();

    // Get a list of top level application elements.
    ApplicationElementSeq aeSeq[] = as.getTopLevelElements("AoAny");

    ...

    // Close the session.
    session.close();

}
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_BASETYPE](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadApplicationStructure-
Elements.run()
test.highqsoft.avalon.ApplicationStructureTest.testApplicationStructureElements()
test.highqsoft.odsapi.CreateInstanceElementTest.testCreateInstancesTopLevel()
test.highqsoft.odsapi.XATFCopyTest.testCopyAllInstances()
```

4.7.16 listElements of interface ApplicationStructure**Purpose:**

Get the names of the application elements that match the pattern. The pattern is case sensitive and may contain wildcard characters.

Return:

[NameSequence](#) applElemNames *The names of the application elements.*

Parameter:

[Pattern](#) aePattern (in) *The name or the search pattern for the requested base elements.*

Java Calling Sequence:

```
Name[] applElemNames = applStruct.listElements(aePattern);
```



Java Example:

```

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 structure.
    ApplicationStructure as = session.getApplicationStructure();

    // Get a list of application element names.
    String aeNameList[] = as.listElements("*");

    ...

    // Close the session.
    session.close();

}

```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.7.17 listElementsByBaseType of interface ApplicationStructure**Purpose:**

Get the names of application elements that are derived from the given base type.

Return:

[NameSequence](#) applElemNames *The names of the application elements.*

Parameter:

[BaseType](#) aeType (in) *The requested base type. The base element type can be a pattern.*

Java Calling Sequence:

```
Name[] applElemNames = applStruct.listElementsByBaseType(aeType);
```

Java Example:

```

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 structure.
    ApplicationStructure as = session.getApplicationStructure();

```

```

    // Get a list of application element names.
    String aeNameList[] = as.listElementsByBaseType("AoTest");

    ...

    // Close the session.
    session.close();
}

```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_BASETYPE](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.7.18 listEnumerations of interface ApplicationStructure**Purpose:**

Get the list of all enumeration names.

Return:

[NameSequence](#) enumNames *List with all enumeration names.*

Parameter:

None.

Since:

ASAM ODS 5.0

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadTestEnumeration-
Definition.run()
test.highqsoft.avalon.ApplicationStructureTest.testEnumerationDefinition()
test.highqsoft.odsapi.EnumerationTest.testCreateEnumerationDefinition()

```

4.7.19 listTopLevelElements of interface ApplicationStructure**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.



Return:

[NameSequence](#) applElemNames *The names of the application elements.*

Parameter:

[BaseType](#) aeType (in) *The requested base type.*

Java Calling Sequence:

```
Name[] applElemNames = applStruct.listTopLevelElements(aeType);
```

Java Example:

```
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 structure.
    ApplicationStructure as = session.getApplicationStructure();

    // Get a list of top level application element names.
    String aeNameList[] = as.listTopLevelElements("*");

    ...

    // Close the session.
    session.close();
}
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_BASETYPE](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadApplicationStructure-
Elements.run()
test.highqsoft.avalon.ApplicationStructureTest.testApplicationStructureElements()
```

4.7.20 removeElement of interface ApplicationStructure**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.

Note:

It is only allowed to modify the application structure if the application element has no instances.

Return:

void

Parameter:

[ApplicationElement](#) applElem (in) *The application element to be removed.*

Java Calling Sequence:

```
applStruct.removeElement(applElem);
```

Java Example:

```
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 structure.
    ApplicationStructure as = session.getApplicationStructure();
    // Get base structure.
    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 structure.
    as.removeElement(ae);
    ...
    // Close the session.
    session.close();
}
```

Errors:

[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](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

4.7.21 removeEnumerationDefinition of interface ApplicationStructure

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.

Note:

See also **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

[T_STRING](#) enumName (in) *Name of the enumeration to remove.*

Java Calling Sequence:

```
asObj.removeEnumerationDefinition(enumName);
```

Since:

ASAM ODS 5.0

Errors:

[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](#)

4.7.22 removeRelation of interface ApplicationStructure

Purpose:

This method removes an application relation from the 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.

Note:

It is only allowed to modify the application structure if the involved application elements have no instances.

Return:

void

Parameter:

[ApplicationRelation](#) applRel (in) *The application relation to be removed.*

Java Calling Sequence:

```
applStruct.removeRelation(applRel);
```

Java Example:

```
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 structure.
    ApplicationStructure as = session.getApplicationStructure();
    // Create a new application relation.
    ApplicationRelation ar = as.createRelation();
    ...
    // Remove an application relation from an application structure.
    as.removeRelation(ar);
    ...
    // Close the session.
    session.close();
}

```

Errors:

[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](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

4.8 BaseAttribute

Package:

org.asam.ods

Description:

The ASAM ODS base attribute interface.

4.8.1 getBaseElement of interface BaseAttribute

Purpose:

Return the base element to which the attribute belongs..

Return:

[BaseElement](#) baseElem *The base element of the attribute.*

Parameter:

None.

Java Calling Sequence:

```
BaseElement baseElem = baseAttr.getBaseElement();
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)



Tested by:

test.hiqsoft.odsapi.ApplModelCreateTest.testCheckElement()

4.8.2 `getDataType` of interface BaseAttribute

Purpose:

Get the data type of the base attribute.

Return:

[DataType](#) baDataType *The data type of the base attribute.*

Parameter:

None.

Java Calling Sequence:

```
DataType baDataType = baseAttr.getDataType();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.8.3 `getEnumerationDefinition` of interface BaseAttribute

Purpose:

Get the definition of the enumeration of the base attribute.

Return:

[EnumerationDefinition](#) enumDef *The ASAM ODS enumeration.*

Parameter:

None.

Java Calling Sequence:

```
EnumerationDefinition enumDef = aaObj.getEnumerationDefinition();
```

Since:

ASAM ODS 5.1

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_INVALID_DATATYPE](#)

4.8.4 getName of interface BaseAttribute

Purpose:

Get the name of the base attribute.

Return:

[Name](#) baName *The name of the base attribute.*

Parameter:

None.

Java Calling Sequence:

```
Name baName = baseAttr.getName();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.createNewInstance()
test.highqsoft.odsapi.security.InstanceCheckTest.testSubTestRights()
test.highqsoft.odsapi.ApplModelCreateTest.testCheckElement()
test.highqsoft.odsapi.AthosTestCase.getWriteableAttributes()
test.highqsoft.odsapi.CreateInstanceElementTest.testAeCreateInstances()
test.highqsoft.odsapi.CreateInstanceTestCase.newInstance()
test.highqsoft.odsapi.InstanceAttributeTest.testManyInstanceAttributeDatatype()
test.highqsoft.odsapi.InstanceAttributeTest.testSetValue()
test.highqsoft.odsapi.InstanceAttributeTest.testSetValueSeq()
test.highqsoft.odsapi.InstanceAttributeTest.testSingleInstanceAttributeDatatype()
....
```

4.8.5 isObligatory of interface BaseAttribute

Purpose:

Get the obligatory flag of the base attribute.

Return:

[T_BOOLEAN](#) baIsObligatory *The obligatory flag of the base attribute.*

Parameter:

None.

Java Calling Sequence:

```
T_BOOLEAN baIsObligatory = baseAttr.isObligatory();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.8.6 isUnique of interface BaseAttribute

Purpose:

Get the unique flag of the base attribute.

Return:

`T_BOOLEAN` baIsUnique *The unique flag of the base attribute.*

Parameter:

None.

Java Calling Sequence:

```
T_BOOLEAN baIsUnique = baseAttr.isUnique();
```

Errors:

`AO_CONNECTION_LOST`
`AO_IMPLEMENTATION_PROBLEM`
`AO_NOT_IMPLEMENTED`
`AO_NO_MEMORY`
`AO_SESSION_NOT_ACTIVE`

Tested by:

test.highqsoft.odsapi.ApplModelCreateTest.testCheckElement()

4.9 BaseElement

Package:

org.asam.ods

Description:

The ASAM ODS base element interface.

4.9.1 getAllRelations of interface BaseElement

Purpose:

Get all known relations of the base element.

Return:

`BaseRelationSequence` baseRels *All known relations of the base element.*

Parameter:

None.

Java Calling Sequence:

```
BaseRelation[] baseRels = baseElem.getAllRelations();
```

Errors:

`AO_CONNECTION_LOST`
`AO_IMPLEMENTATION_PROBLEM`
`AO_NOT_IMPLEMENTED`
`AO_NO_MEMORY`
`AO_SESSION_NOT_ACTIVE`

Tested by:

test.highqsoft.avalon.ApplicationRelationTest.testApplicationRelationBase()

4.9.2 getAttributes of interface BaseElement**Purpose:**

Get attributes of the base element.

Return:

[BaseAttributeSequence](#) baseAttrs *The requested attributes of the base element.*

Parameter:

[Pattern](#) baPattern (in) *The name or the search pattern for the requested base attributes.*

Java Calling Sequence:

```
BaseAttribute[] baseAttrs = baseElem.getAttributes(baPattern);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.ApplicationStructureTest.testApplicationStructureValue()
 test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testAddAttribute()
 test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
 test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
 test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateElement()
 test.highqsoft.odsapi.ApplModelCreateTest.testSetBaseAttr()

4.9.3 getRelatedElementsByRelationship of interface BaseElement**Purpose:**

Get the related elements of a base element defined by the relationship.

Return:

[BaseElementSequence](#) baseElems *The related elements of a base element.*

Parameter:

[Relationship](#) brRelationship (in) *The requested relationship.*

Java Calling Sequence:

```
BaseElement[] baseElems = baseElem.getRelatedElementsByRelationship(brRelationship);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)



AO_IMPLEMENTATION_PROBLEM
AO_INVALID_RELATIONSHIP
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

4.9.4 getRelationsByType of interface BaseElement

Purpose:

Get the base element's relations of the requested relation type.

Return:

[BaseRelationSequence](#) baseRels *The base element's relations of the requested type.*

Parameter:

[RelationType](#) brRelationType (in) *The requested relation type.*

Java Calling Sequence:

```
BaseRelation[] baseRels = baseElem.getRelationsByType(brRelationType);
```

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_INVALID_RELATION_TYPE
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

```
test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadApplicationStructure-  
Base.run()  
test.highqsoft.avalon.ApplicationStructureTest.testApplicationStructureBase()
```

4.9.5 getType of interface BaseElement

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.

Return:

[BaseType](#) beType *The type of the base element.*

Parameter:

None.

Java Calling Sequence:

```
BaseType beType = baseElem.getType();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.BaseStructureTest.testBaseRelations()
 test.highqsoft.odsapi.ApplModelCreateTest.testCheckElement()
 test.highqsoft.odsapi.AthosTestCase.checkBeToIgnore()
 test.highqsoft.odsapi.CreateInstanceElementTest.testCreateInstancesTopLevel()
 test.highqsoft.odsapi.XATFCopyTest.compareElement()

4.9.6 isTopLevel of interface BaseElement

Purpose:

Get whether or not the base element is a top level element. Top level elements are elements without a father.

Return:

[T_BOOLEAN](#) beIsTopLevel *Boolean whether or not the base element is a top level element.*

Parameter:

None.

Java Calling Sequence:

```
T_BOOLEAN beIsTopLevel = baseElem.isTopLevel();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.9.7 listAttributes of interface BaseElement

Purpose:

Get attribute names of the base element.

Return:

[NameSequence](#) baseElemNames *The requested attribute names of the base element.*

Parameter:

[Pattern](#) baPattern (in) *The name or the search pattern for the requested base attribute names.*

Java Calling Sequence:

```
Name[] baseElemNames = baseElem.listAttributes(baPattern);
```



Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

`test.highqsoft.avalon.BaseStructureTest.testBaseElement()`

4.9.8 listRelatedElementsByRelationship of interface BaseElement**Purpose:**

Get the related element names of the base element defined by the relationship.

Return:

[BaseTypeSequence](#) baseTypes *The related element names of the base element.*

Parameter:

[Relationship](#) brRelationship (in) *The requested relationship.*

Java Calling Sequence:

```
BaseType[] baseTypes = baseElem.listRelatedElementsByRelationship(brRelationship);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_RELATIONSHIP](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.10 BaseRelation**Package:**

`org.asam.ods`

Description:

The ASAM ODS base relation interface.

4.10.1 getElem1 of interface BaseRelation**Purpose:**

Get the first base element of the base relation.

Return:

[BaseElement](#) baseElem *The first base element of the base relation.*

Parameter:

None.

Java Calling Sequence:

```
BaseElement baseElem = baseRel.getElem1();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.10.2 getElem2 of interface BaseRelation

Purpose:

Get the second base element of the base relation.

Return:

[BaseElement](#) baseElem *The second base element of the base relation.*

Parameter:

None.

Java Calling Sequence:

```
BaseElement baseElem = baseRel.getElem2();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.10.3 getInverseRelationName of interface BaseRelation

Purpose:

Return the inverse name of the base relation.

Return:

[Name](#) invRelName *The name of the inverse relation.*

Parameter:

None.

Java Calling Sequence:

```
String invRelName = brObj.getInverseRelationName();
```

Since:

ASAM ODS 5.1



Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

test.highqsoft.avalon.BaseStructureTest.testBaseRelations()

4.10.4 getInverseRelationRange of interface BaseRelation

Purpose:

Get the inverse relation range of the base relation.

Return:

[RelationRange](#) brRange *The inverse relation range of the base relation.*

Parameter:

None.

Java Calling Sequence:

```
RelationRange brRange = baseRel.getInverseRelationRange();
```

Errors:

AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

4.10.5 getInverseRelationship of interface BaseRelation

Purpose:

Get the inverse relationship of the base relation.

Return:

[Relationship](#) relationship *The inverse relationship of the base relation.*

Parameter:

None.

Java Calling Sequence:

```
Relationship relationship = baseRel.getInverseRelationship();
```

Errors:

AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

test.highqsoft.avalon.BaseStructureTest.testBaseRelations()

4.10.6 getRelationName of interface BaseRelation

Purpose:

Get the relation name of the base relation.

Return:

[Name](#) brName *The relation name of the base relation.*

Parameter:

None.

Java Calling Sequence:

```
Name brName = baseRel.getRelationName();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadApplicationStructureBase.run()
test.highqsoft.avalon.ApplicationStructureTest.testApplicationStructureBase()
test.highqsoft.avalon.BaseStructureTest.testBaseRelations()

4.10.7 getRelationRange of interface BaseRelation

Purpose:

Get the relation range of the base relation.

Return:

[RelationRange](#) brRange *The relation range of the base relation.*

Parameter:

None.

Java Calling Sequence:

```
RelationRange brRange = baseRel.getRelationRange();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()



4.10.8 getRelationship of interface BaseRelation

Purpose:

Get the relationship of the base relation.

Return:

[Relationship](#) relationship *The relationship of the base relation.*

Parameter:

None.

Java Calling Sequence:

```
Relationship relationship = baseRel.getRelationship();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.10.9 getRelationType of interface BaseRelation

Purpose:

Get the relation type of the base relation.

Return:

[RelationType](#) brType *The relation type of the base relation.*

Parameter:

None.

Java Calling Sequence:

```
RelationType brType = baseRel.getRelationType();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.BaseStructureTest.testBaseRelations()
```

4.11 BaseStructure

Package:

org.asam.ods

Description:

The ASAM ODS base structure interface.

4.11.1 getElementByType of interface BaseStructure

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.

Return:

[BaseElement](#) baseElem *The requested base element.*

Parameter:

[BaseType](#) beType (in) *The name of the requested base element.*

Java Calling Sequence:

```
BaseElement baseElem = baseStruct.getElementByType(beType);
```

Java Example:

```
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 structure.
    BaseStructure bs = session.getBaseStructure();

    // Get a the base element of type AoMeasurement.
    BaseElement be = bs.getElementByType("AoMeasurement");

    ...

    // Close the session.
    session.close();
}
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_BASETYPE](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateElement()
test.highqsoft.odsapi.ApplModelCreateTest.testCreateElements()
test.highqsoft.odsapi.EnumerationTest.testWriteApplicationModel()
```


4.11.2 getElements of interface BaseStructure

Purpose:

Get the base elements that match the pattern. The pattern is case sensitive and may contain wildcard characters.

Return:

[BaseElementSequence](#) baseElems *The requested base elements.*

Parameter:

[Pattern](#) bePattern (in) *The name or the search pattern for the requested base elements.*

Java Calling Sequence:

```
BaseElement[] baseElems = baseStruct.getElements(bePattern);
```

Java Example:

```
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 structure.
    BaseStructure bs = session.getBaseStructure();

    // Get a list of base elements.
    BaseElement beList[] = bs.getElements("*");

    ...

    // Close the session.
    session.close();
}
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.BaseStructureTest.testBaseElement()
test.highqsoft.avalon.BaseStructureTest.testBaseElementTopLevel()
test.highqsoft.avalon.BaseStructureTest.testBaseRelations()
```

4.11.3 getRelation of interface BaseStructure

Purpose:

Get the base relation between two base elements.

Return:

[BaseRelation](#) baseRel *The base relation between the two base elements.*

Parameter:

[BaseElement](#) elem1 (in) *The base element from which the relation starts.*

[BaseElement](#) elem2 (in) *The base element to which the relation points.*

Java Calling Sequence:

```
BaseRelation baseRel = baseStruct.getRelation(elem1,elem2);
```

Java Example:

```
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 structure.
    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();
}
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.BaseStructureTest.testBaseRelations()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateAttrElement()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateChildElement()
test.highqsoft.odsapi.ApplModelCreateTest.testCreateElements()
```

4.11.4 getTopLevelElements of interface BaseStructure**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.

Return:

[BaseElementSequence](#) baseElems *The requested top level base elements.*



Parameter:

Pattern bePattern (in) *The name or the search pattern for the requested top level base elements.*

Java Calling Sequence:

```
BaseElement[] baseElems = baseStruct.getTopLevelElements(bePattern);
```

Java Example:

```
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 structure.
    BaseStructure bs = session.getBaseStructure();

    // Get a list of top level base elements.
    BaseElement beList[] = bs.getTopLevelElements("*");

    ...

    // Close the session.
    session.close();
}
```

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

```
test.highqsoft.avalon.BaseStructureTest.testBaseStructureElements()
```

4.11.5 getVersion of interface BaseStructure**Purpose:**

Get the version of the base model. The version of the base model is the version of the ASAM ODS base model.

Return:

T_STRING version *The version of the ASAM ODS base model.*

Parameter:

None.

Java Calling Sequence:

```
T_STRING version = baseStruct.getVersion();
```

Java Example:

```

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 structure.
    BaseStructure bs = session.getBaseStructure();

    // Get current base structure version.
    String bsVersion = bs.getVersion();

    ...

    // Close the session.
    session.close();

}

```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.BaseStructureTest.testBaseStructureElements()

4.11.6 listElements of interface BaseStructure**Purpose:**

Get the base element names that match the pattern. The pattern is case sensitive and may contain wildcard characters.

Return:

[BaseTypeSequence](#) baseTypes *The requested base element names.*

Parameter:

[Pattern](#) bePattern (in) *The name or the search pattern for the requested base element names.*

Java Calling Sequence:

```
BaseType[] baseTypes = baseStruct.listElements(bePattern);
```

Java Example:

```

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 structure.
    BaseStructure bs = session.getBaseStructure();

    // Get a list of base element names.
    String beNameList[] = bs.ListElements("*");

    ...

    // Close the session.
    session.close();

}

```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.11.7 listTopLevelElements of interface BaseStructure**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.

Return:

[BaseTypeSequence](#) baseTypes *The requested top level base element names.*

Parameter:

[Pattern](#) bePattern (in) *The name or the search pattern for the requested top level base element names.*

Java Calling Sequence:

```
BaseType[] baseTypes = baseStruct.listTopLevelElements(bePattern);
```

Java Example:

```

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 structure.
    BaseStructure bs = session.getBaseStructure();

    // Get a list of top level base element names.
    String beNameList[] = bs.ListTopLevelElements("*");

    ...
}

```

```
// Close the session.  
session.close();  
  
}
```

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

```
test.highqsoft.avalon.BaseStructureTest.testBaseStructureElements()
```

4.12 Blob

Package:

org.asam.ods

Description:

The ASAM ODS blob interface.

4.12.1 append of interface Blob

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.

Return:

void

Parameter:

[S_BYTE](#) value (in) *The byte sequence.*

Java Calling Sequence:

```
blob.append(value);
```

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE



4.12.2 compare of interface Blob

Purpose:

Compares the content of the binary large object. The headers are not compared.

Return:

`T_BOOLEAN` `contentEqual` *A flag whether or not the compared blobs are equal.*

Parameter:

`T_BLOB` `aBlob` (in) *The blob to compare.*

Java Calling Sequence:

```
T_BOOLEAN contentEqual = blob.compare(aBlob);
```

Errors:

`AO_BAD_PARAMETER`
`AO_CONNECTION_LOST`
`AO_IMPLEMENTATION_PROBLEM`
`AO_NOT_IMPLEMENTED`
`AO_NO_MEMORY`
`AO_SESSION_NOT_ACTIVE`

Tested by:

```
test.hiqsoft.odsapi.UpdateInstanceElementTest.testBlobCompare()
```

4.12.3 destroy of interface Blob

Purpose:

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.

Return:

`void`

Parameter:

None.

Java Calling Sequence:

```
blob.destroy()
```

Since:

ASAM ODS 5.0

Errors:

`AO_BAD_PARAMETER`
`AO_CONNECTION_LOST`
`AO_IMPLEMENTATION_PROBLEM`
`AO_NOT_IMPLEMENTED`
`AO_NO_MEMORY`
`AO_SESSION_NOT_ACTIVE`

4.12.4 get of interface Blob

Purpose:

Get a part of the binary large object.

Return:

[S_BYTE](#) `byteStream` *The request part of the blob data.*

Parameter:

[T_LONG](#) `offset` (in) *The starting position of the data in the blob.*

[T_LONG](#) `length` (in) *The number of bytes requested from the blob.*

Java Calling Sequence:

```
S_BYTE byteStream = blob.get(offset,length);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.read.AttributeValueTest.testBlobAttributeValues()
```

4.12.5 getHeader of interface Blob

Purpose:

Get the header of the binary large object.

Return:

[T_STRING](#) `header` *The blob header.*

Parameter:

None.

Java Calling Sequence:

```
T_STRING header = blob.getHeader();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.12.6 getLength of interface Blob

Purpose:

Get the length of the binary large object without loading it.

Return:

`T_LONG` length *The blob length.*

Parameter:

None.

Java Calling Sequence:

```
T_LONG length = blob.getLength();
```

Errors:

`AO_CONNECTION_LOST`
`AO_IMPLEMENTATION_PROBLEM`
`AO_NOT_IMPLEMENTED`
`AO_NO_MEMORY`
`AO_SESSION_NOT_ACTIVE`

4.12.7 set of interface Blob

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.

Return:

void

Parameter:

`S_BYTE` value (in) *The new blob data.*

Java Calling Sequence:

```
blob.set(value);
```

Errors:

`AO_BAD_PARAMETER`
`AO_CONNECTION_LOST`
`AO_IMPLEMENTATION_PROBLEM`
`AO_NOT_IMPLEMENTED`
`AO_NO_MEMORY`
`AO_SESSION_NOT_ACTIVE`

Tested by:

```
test.hiqsoft.odsapi.UpdateInstanceElementTest.testBlobCompare()
```

4.12.8 setHeader of interface Blob

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.

Return:

void

Parameter:

[T_STRING](#) header (in) *The blob header.*

Java Calling Sequence:

```
blob.setHeader(header);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.13 Column

Package:

org.asam.ods

Description:

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.

Note:

The name of the column is the name of the instance of the measurement quantity. Due to the fact ASAM ODS requires the name of the local column must be identical with the name of the measurement quantity there will be normally no difference.

4.13.1 destroy of interface Column

Purpose:

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.

Note:

This method have only influence when corba is used.



Return:

void

Parameter:

None.

Java Calling Sequence:`column.destroy()`**Since:**

ASAM ODS 5.0

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

`test.highqsoft.avalon.MultiSessionTest.threadCompareValues.run()`
`test.highqsoft.avalon.ValueMatrixTest.testCheckUnit()`
`test.highqsoft.avalon.ValueMatrixTest.testCompareValues()`
`test.highqsoft.avalon.ValueMatrixTest.testValueMatrixGetValueVector()`
`test.highqsoft.avalon.ValueMatrixTest.testValueMatrixTime()`
`test.highqsoft.odsapi.read.MeasurementTest.testValueMatrixPoint()`
`test.highqsoft.odsapi.XATFCopyTest.testCompareValueMatrixValues()`

4.13.2 `getDataType` of interface `Column`

Purpose:

Get the data type of the column.

Note:

Loading of the datatype of the localcolumn in case the datatype isn't stored at the measurement quantity.

Return:[DataType](#) *dataType* *The data type of the column.***Parameter:**

None.

Java Calling Sequence:`DataType dataType = column.getDataType();`**Since:**

ASAM ODS 5.0

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)

AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

4.13.3 getFormula of interface Column

Purpose:

Get the formula of the column.

There is no formula defined in ASAM ODS, so there is nothing to return.

Note:

Formula handling not implemented in ATHOS.

Return:

T_STRING formula *The formula of the column.*

Parameter:

None.

Java Calling Sequence:

```
T_STRING formula = column.getFormula();
```

Errors:

AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

4.13.4 getName of interface Column

Purpose:

Get the name of the column.

Note:

The name of the column is the name of the instance of the measurement quantity. Due to the fact ASAM ODS requires the name of the local column must be identical with the name of the measurement quantity there will be normally no difference.

Return:

Name columnName *The name of the column.*

Parameter:

None.

Java Calling Sequence:

```
Name columnName = column.getName();
```



Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.MultiSessionTest.threadCreateValueMatrixFromSm.run()  
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromSm()  
test.highqsoft.odsapi.read.MeasurementTest.testValueMatrixPoint()  
test.highqsoft.odsapi.CreateInstanceTestCase.updateMeasurementInstances()  
test.highqsoft.odsapi.UpdateMeasurementTest.testCheckMeasurement()  
test.highqsoft.odsapi.XATFCopyTest.testCompareValueMatrixValues()
```

4.13.5 getSourceMQ of interface Column

Purpose:

Get the source measurement quantity.

Return:

[InstanceElement](#) instElem *The source measurement quantity.*

Parameter:

None.

Java Calling Sequence:

```
InstanceElement instElem = column.getSourceMQ();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.ValueMatrixTest.testCompareColumns()  
test.highqsoft.avalon.ValueMatrixTest.testUnitConversion()
```

4.13.6 getUnit of interface Column

Purpose:

Get the unit of the column.

Return:

[T_STRING](#) unit *The unit of the column.*

Parameter:

None.

Java Calling Sequence:

```
T_STRING unit = column.getUnit();
```

Errors:

```
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
```

Tested by:

```
test.highqsoft.avalon.ValueMatrixTest.testCheckUnit()
test.highqsoft.odsapi.read.MeasurementTest.testValueMatrixUnit()
```

4.13.7 isIndependent of interface Column**Purpose:**

Is the column an independent column

Return:

T_BOOLEAN independent *The independent flag of the column.*

Parameter:

None.

Java Calling Sequence:

```
T_BOOLEAN independent = column.isIndependent();
```

Errors:

```
AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
```

4.13.8 isScaling of interface Column**Purpose:**

Is the column an scaling column

Return:

T_BOOLEAN scaling *Tells if the column is a scaling column.*

Parameter:

None.

Java Calling Sequence:

```
T_BOOLEAN scaling = column.isScaling();
```



Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

4.13.9 setFormula of interface Column**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.

There is no formula in ASAM ODS so don't try to set the formula.

Return:

void

Parameter:

T_STRING formula (in) *The formula.*

Java Calling Sequence:

```
column.setFormula(formula);
```

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

4.13.10 setIndependent of interface Column**Purpose:**

Set the column as an independent column.

Return:

void

Parameter:

T_BOOLEAN independent (in) *The new value of the independent flag.*

Java Calling Sequence:

```
column.setIndependent(independent);
```

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM

AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
AO_TRANSACTION_NOT_ACTIVE

4.13.11 setScaling of interface Column

Purpose:

Set the column to a scaling column.

Return:

void

Parameter:

T_BOOLEAN scaling (in) *The new value of the scaling flag.*

Java Calling Sequence:

```
column.setScaling(scaling);
```

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
AO_TRANSACTION_NOT_ACTIVE

4.13.12 setUnit of interface Column

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.

Return:

void

Parameter:

T_STRING unit (in) *The physical unit.*

Java Calling Sequence:

```
column.setUnit(unit);
```

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

test.hiqsoft.avalon.ValueMatrixTest.testUnitConversion()



4.14 ElemResultSetExtSeqIterator

Package:

org.asam.ods

Description:

For iteration through the result elements, there is also a new iterator necessary called ElemResultSetExtSeqIterator. Its 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.

4.14.1 destroy of interface ElemResultSetExtSeqIterator

Purpose:

Destroy the iterator and free the associated memory.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
ersIter.destroy();
```

Errors:

AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

org.asam.ods.ElemResultSetExtSeqIteratorPOA._invoke()

4.14.2 getCount of interface ElemResultSetExtSeqIterator

Purpose:

Get the total number of elements accessible by the iterator.

Return:

T_LONG count *The number of elements accessible by the iterator.*

Parameter:

None.

Java Calling Sequence:

```
T_LONG cnt = ersIter.getCount();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.14.3 nextN of interface ElemResultSetExtSeqIterator**Purpose:**

Get the next n elements from the sequence.

Return:

[ElemResultSetExtSequence](#) elemResults *The next n attribute values from the element result set sequence.*

Parameter:

[T_LONG](#) how_many (in) *The number of requested elements.*

Java Calling Sequence:

```
T_LONG how_many = 10;
ElemResultSetExt[] elemResults = ersIter.nextN(how_many)
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.14.4 nextOne of interface ElemResultSetExtSeqIterator**Purpose:**

Get the next element from the sequence.

Return:

[ElemResultSetExt](#) elemResult *The next attribute values from the element result set sequence.*

Parameter:

None.

Java Calling Sequence:

```
ElemResultSetExt elemResult = ersIter.nextOne()
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.14.5 reset of interface ElemResultSetExtSeqIterator

Purpose:

Reset the pointer in the element sequence to the first element.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
ersIter.reset();
```

Errors:

AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

4.15 EnumerationDefinition

Package:

org.asam.ods

Description:

The ASAM ODS enumeration interface.

4.15.1 addItem of interface EnumerationDefinition

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. Current server implementations restrict it to 128 characters.

Return:

void

Parameter:

T_STRING itemName (in) *The name of the new item.*

Java Calling Sequence:

```
enumDef.addItem(itemName);
```

Errors:

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

Tested by:

test.highqsoft.odsapi.EnumerationTest.testWriteApplicationModel()

4.15.2 getIndex of interface EnumerationDefinition**Purpose:**

Get the index of the enumeration.

Return:

T_LONG enumIndex *The index of the enumeration.*

Parameter:

None.

Java Calling Sequence:

```
T_LONG enumIndex = enumDef.getIndex();
```

Errors:

AO_BAD_PARAMETER
 AO_CONNECTION_LOST
 AO_IMPLEMENTATION_PROBLEM
 AO_NOT_IMPLEMENTED
 AO_NO_MEMORY
 AO_SESSION_NOT_ACTIVE

4.15.3 getItem of interface EnumerationDefinition**Purpose:**

Get the value of an item.

Return:

T_LONG item *The number of the item.*

Parameter:

T_STRING itemName (in) *The name of the item.*

Java Calling Sequence:

```
T_LONG item = enumDef.getItem(itemName);
```

Errors:

AO_BAD_PARAMETER
 AO_CONNECTION_LOST
 AO_IMPLEMENTATION_PROBLEM
 AO_NOT_IMPLEMENTED
 AO_NO_MEMORY
 AO_SESSION_NOT_ACTIVE
 AO_NOT_FOUND



Tested by:

test.hiqsoft.odsapi.EnumerationTest.testWriteEnumInstance()

4.15.4 getItemName of interface EnumerationDefinition**Purpose:**

Get the name of an item.

Return:

`T_STRING` itemName *The name of the item.*

Parameter:

`T_LONG` item (in) *The value of the item.*

Java Calling Sequence:

```
T_STRING itemName = enumDef.getItemName(item);
```

Errors:

`AO_BAD_PARAMETER`
`AO_CONNECTION_LOST`
`AO_IMPLEMENTATION_PROBLEM`
`AO_NOT_IMPLEMENTED`
`AO_NO_MEMORY`
`AO_SESSION_NOT_ACTIVE`
`AO_NOT_FOUND`

Tested by:

test.hiqsoft.odsapi.EnumerationTest.testReadEnumInstance()

4.15.5 getName of interface EnumerationDefinition**Purpose:**

Get the name of the enumeration.

Return:

`T_STRING` enumName *Name of the enumeration.*

Parameter:

None.

Java Calling Sequence:

```
T_STRING enumName = enumDef.getName();
```

Errors:

`AO_BAD_PARAMETER`
`AO_CONNECTION_LOST`
`AO_IMPLEMENTATION_PROBLEM`
`AO_NOT_IMPLEMENTED`
`AO_NO_MEMORY`
`AO_SESSION_NOT_ACTIVE`

Tested by:

```
test.highqsoft.avalon.ApplicationStructureMultiSessionTest.threadTestEnumeration-
Definition.run()
test.highqsoft.avalon.ApplicationStructureTest.testEnumerationDefinition()
test.highqsoft.odsapi.EnumerationTest.testReadEnumInstance()
test.highqsoft.odsapi.EnumerationTest.testWriteEnumInstance()
```

4.15.6 listItemNames of interface EnumerationDefinition**Purpose:**

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.

Return:

[NameSequence](#) nameList *List with all possible names of the enumeration items.*

Parameter:

None.

Java Calling Sequence:

```
NameSequence nameList = enumDef.listItemNames();
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.15.7 renameItem of interface EnumerationDefinition**Purpose:**

Rename the item of the enumeration. This method modifies the application model and is only allowed for the superuser.

Return:

void

Parameter:

[T_STRING](#) oldItemName (in) *The existing name of the item.*

[T_STRING](#) newItemName (in) *the new name of the item.*

Java Calling Sequence:

```
enumDef.renameItem(oldItemName, newItemName);
```

Errors:

[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

4.15.8 setName of interface EnumerationDefinition

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. Current server implementations restrict it to 30 characters.

Return:

void

Parameter:

T_STRING enumName (in) *Name of the enumeration.*

Java Calling Sequence:

```
enumDef.setName(enumName);
```

Errors:

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

4.16 InstanceElement

Package:

org.asam.ods

Description:

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.

4.16.1 addInstanceAttribute of interface InstanceElement

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.

Return:

void

Parameter:[NameValueUnit](#) instAttr (in) *The instance attribute to be added.***Java Calling Sequence:**

```
instElem.addInstanceAttribute(instAttr);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.16.2 compare of interface InstanceElement

Purpose:

Compare two instance elements. The Id's of the application elements and the Id's of the instance elements are compared. The Id's of the application elements will be compare first.

Return:

[T_LONGLONG](#) diff *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.

Parameter:[InstanceElement](#) compIeObj (in) *The instance element to compare with.***Java Calling Sequence:**

```
T_LONGLONG diff = ieObj.compare(compIeObj);
```

Since:

ASAM ODS 5.0

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.WriteInstanceElementTest.testCopyTest()
```


4.16.3 createRelatedInstances of interface InstanceElement

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.

Return:

[InstanceElementSequence](#) elemList *The list with the new created instances.*

Parameter:

[ApplicationRelation](#) applRel (in) *The application relation for wich the related instances will be created.*

[NameValueSeqUnitSequence](#) attributes (in) *The attributes of the new created instances.*

[ApplicationRelationInstanceElementSeqSequence](#) relatedInstances (in) *The list with related instances for different application relations.*

Java Calling Sequence:

```
InstanceElement ieObj;
ApplicationRelation applRel;
NameValueSeqUnit attributes[];
ApplicationjRealtionInstanceElementSeq relatedInstances[]
InstanceElement elemList[] = ieObj.createRelatedInstances(applRel, attributes, relatedInstances);
```

Since:

ASAM ODS 5.0

Errors:

[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](#)

4.16.4 createRelation of interface InstanceElement

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.

Return:

void

Parameter:

[ApplicationRelation](#) relation (in) *The application relation.*

[InstanceElement](#) instElem (in) *The instance element.*

Java Calling Sequence:

```
instElem.createRelation(relation,instElem);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_RELATION](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testIeCreateRelatedInstances()
test.highqsoft.odsapi.read.AsamPathTest.testCreateAsamPathSpaces()
test.highqsoft.odsapi.read.InstanceRelationTest.testCreateRecursiveInstances()
test.highqsoft.odsapi.read.ExtendedQueryTest.testCreateTestDataN_M()
test.highqsoft.odsapi.read.ExtendedQueryTest.testCreateTestDataOuterJoin()
test.highqsoft.odsapi.CreateInstanceElementTest.createRelationCheck()
test.highqsoft.odsapi.CreateInstanceTestCase.createLcInst()
test.highqsoft.odsapi.CreateInstanceTestCase.createMeasurementInstances()
test.highqsoft.odsapi.CreateInstanceTestCase.createMeqQtyRelation()
test.highqsoft.odsapi.CreateInstanceTestCase.createNMRelation()
....
```

4.16.5 deepCopy of interface InstanceElement

Purpose:

Provides an easy-to-use and effective copy mechanism 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.

Note:

All the child instances are copied also the measurement quantities, submatrixes and localcolumn.

Return:

[InstanceElement](#) newInstElem *The reference to the copied instance element hierarchy.*

Parameter:

[T_STRING](#) newName (in) *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.*



T_STRING newVersion (in) *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);
```

Since:

ASAM ODS 4.1

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```
test.hightsoft.odsapi.WriteInstanceElementTest.testCopyTest()
```

4.16.6 destroy of interface InstanceElement

Purpose:

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.

Note:

This method have only influence when corba is used.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
ieObj.destroy()
```

Java Example:

```
// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

// Get ApplElemAccess Interface
ApplElemAccess aea = aoSession.getApplElemAccess();

// Get the all application elements.
ApplicationElement aeList[] = as.getElements("*");

for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {
    // Name of the application element
    String aeName = aeList[aeIndex].getName();

    // Get the instances of the application element
```

```

InstanceElement ieObj;
InstanceElementIterator iei = aeList[aeIndex].getInstances("*");
if (iei != null) {
    do {
        ieObj = iei.nextOne();
        if (ieObj != null) {
            // Destroy the instance element, the object is not
            // used anymore at the client.
            ieObj.destroy();
        }
    } while (ieObj != null);
    // Destroy the instance iterator, the object is not
    // used anymore at the client.
    iei.destroy();
}
} // for all application elements.

```

Since:

ASAM ODS 5.0

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.InstanceElementTest.testCheckRelLccMeq()
test.highqsoft.avalon.InstanceElementTest.testCorbaReferences()
test.highqsoft.avalon.InstanceElementTest.testInstanceRelatedAgainstInverseRelated()
test.highqsoft.avalon.InstanceElementTest.testInstancesNamesQuery()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationApplElemAccess()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationChild()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationNames()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelations()
test.highqsoft.avalon.MeasurementTest.testManyMeasurement()
test.highqsoft.avalon.MeasurementTest.testMeasurement()
....

```

4.16.7 getApplicationElement of interface InstanceElement

Purpose:

Get the application element of the instance element.

Return:

[ApplicationElement](#) applElem *The application element from which the instance element is derived.*

Parameter:

None.

Java Calling Sequence:

```
ApplicationElement applElem = instElem.getApplicationElement();
```



Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.InstanceElementTest.testInstancesRelationChild()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationNames()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelations()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelations.run()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationChild.run()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationNames.run()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromSmSelectMea()
test.highqsoft.avalon.ValueMatrixTest.threadHandleColumn.run()
test.highqsoft.odsapi.DeleteInstanceElementTest.doubleDelete()

```

4.16.8 getAsamPath of interface InstanceElement**Purpose:**

Get the ASAM-Path of the instance element.

Return:

[Name](#) asamPath *The ASAM path to the instance element.*

Parameter:

None.

Java Calling Sequence:

```
Name asamPath = instElem.getAsamPath();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.InstanceElementTest.testCheckRelLccMeq()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationApplElemAccess()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelations()
test.highqsoft.avalon.MultiSessionTest.threadCompareValues.run()
test.highqsoft.avalon.MultiSessionTest.threadCreateValueMatrix.run()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelations.run()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationApplElemAccess.run()
test.highqsoft.avalon.ValueMatrixTest.testCheckUnit()
test.highqsoft.avalon.ValueMatrixTest.testCompareColumns()
test.highqsoft.avalon.ValueMatrixTest.testCompareValues()
...

```

4.16.9 getId of interface InstanceElement

Purpose:

Get the Id of the instance element.

Return:

`T_LONGLONG` ieId *The Id of the instance element.*

Parameter:

None.

Java Calling Sequence:

```
T_LONGLONG ieId = instElem.getId();
```

Java Example:

```
// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

// Get ApplElemAccess Interface
ApplElemAccess aea = aoSession.getApplElemAccess();

/* Get the all application elements. */
ApplicationElement aeList[] = as.getElements("*");

for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {

    // Name of the application element
    String aeName = aeList[aeIndex].getName();

    // Get the instances of the application element
    InstanceElement ieObj;
    InstanceElementIterator iei = aeList[aeIndex].getInstances("*");
    if (iei != null) {
        do {
            ieObj = iei.nextOne();
            if (ieObj != null) {
                // Id of the instance,
                T_LONGLONG iid = ieObj.getId();
                System.out.println(aeName + ", " + iid.low);
                // Destroy the instance element, the object is not
                // used anymore at the client.
                ieObj.destroy();
            }
        } while (ieObj != null);
        // Destroy the instance iterator, the object is not
        // used anymore at the client.
        iei.destroy();
    }
} // for all application elements.
```

Errors:

`AO_CONNECTION_LOST`
`AO_IMPLEMENTATION_PROBLEM`
`AO_NOT_IMPLEMENTED`
`AO_NO_MEMORY`
`AO_SESSION_NOT_ACTIVE`

Tested by:

```
test.highqsoft.avalon.InstanceElementTest.testCorbaReferences()
test.highqsoft.avalon.InstanceElementTest.testInstancesAsamPath()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationApplElemAccess()
```



```

test.highqsoft.avalon.InstanceElementTest.testInstancesRelations()
test.highqsoft.avalon.MultiSessionTest.threadInstancesAsamPath.run()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelations.run()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationAppElemAccess.run()
test.highqsoft.avalon.SecurityTest.testIniRights()
test.highqsoft.avalon.SecurityTest.testRights()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromSmSelectMea()
....

```

4.16.10 getInitialRights of interface InstanceElement

Purpose:

Retrieve access control list information for the initial rights of the given object.

Return:

[InitialRightSequence](#) initialRights *The access control list entries with the initial rights of the given application element.*

Parameter:

None.

Java Calling Sequence:

```
InitialRight[] initialRights = instElem.getInitialRights();
```

Java Example:

```

// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

// Get AppElemAccess Interface
AppElemAccess aea = aoSession.getAppElemAccess();

/* Get the all application elements. */
ApplicationElement aeList[] = as.getElements("");

for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {
    InitialRight [] irSeq;      // The Initial rights.

    // Name of the application element
    String aeName = aeList[aeIndex].getName();

    // Get the instances of the application element, a query
    // for only the Id of the instances using the AppElemAccess
    // interface, would do it also, but the query is harder to
    // code and will blowup the example.
    InstanceElement ieObj;
    InstanceElementIterator iei = aeList[aeIndex].getInstances("");
    if (iei != null) {
        do {
            ieObj = iei.nextOne();
            if (ieObj != null) {
                // Id of the instance,
                T_LONGLONG iid = ieObj.getId();
                // Get the access control list
                irSeq = ieObj.getInitialRights();
                for (int irIndex = 0;
                    irIndex < irSeq.length;
                    irIndex++) {
                    System.out.println(aeName + ", " + iid.low + ", " +
                        irSeq[irIndex].usergroupId.low + ", " +

```

```

        irSeq[irIndex].refAid.low + ", " +
        irSeq[irIndex].rights);
    }
    // Destroy the instance element, the object is not
    // used anymore at the client.
    ieObj.destroy();
}
} while (ieObj != null);
// Destroy the instance iterator, the object is not
// used anymore at the client.
iei.destroy();
}
} // for all application elements.

```

See also:

[getInstanceInitialRights](#) of interface [ApplElemAccess](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.SecurityTest.testIniRights()

4.16.11 getName of interface InstanceElement

Purpose:

Get the name of the instance element.

Return:

[Name](#) ieName *The name of the instance element.*

Parameter:

None.

Java Calling Sequence:

```
Name ieName = instElem.getName();
```

Java Example:

```

// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

// Get ApplElemAccess Interface
ApplElemAccess aea = aoSession.getApplElemAccess();

// Get the all application elements.
ApplicationElement aeList[] = as.getElements("*");

for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {
    // Name of the application element
    String aeName = aeList[aeIndex].getName();

    // Get the instances of the application element

```



```

InstanceElement ieObj;
InstanceElementIterator iei = aeList[aeIndex].getInstances("*");
if (iei != null) {
    do {
        ieObj = iei.nextOne();
        if (ieObj != null) {
            System.out.println(aeName + ", " + ieObj.getName());
            // Destroy the instance element, the object is not
            // used anymore at the client.
            ieObj.destroy();
        }
    } while (ieObj != null);
    // Destroy the instance iterator, the object is not
    // used anymore at the client.
    iei.destroy();
}
} // for all application elements.

```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.InstanceElementTest.testInstancesNames()
test.highqsoft.avalon.InstanceElementTest.testInstancesNamesQuery()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationChild()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationNames()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelations()
test.highqsoft.avalon.MultiSessionTest.threadCompareValues.run()
test.highqsoft.avalon.MultiSessionTest.threadCreateValueMatrixFromSm.run()
test.highqsoft.avalon.MultiSessionTest.threadInstancesNamesQuery.run()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelations.run()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationChild.run()
....

```

4.16.12 getRelatedInstances of interface InstanceElement**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.

Return:

[InstanceElementIterator](#) ieIterator *The related instances.*

Parameter:

[ApplicationRelation](#) applRel (in) *The application relation.*

[Pattern](#) iePattern (in) *The name or the search pattern for the related instance names.*

Java Calling Sequence:

```
InstanceElementIterator ieIterator = instElem.getRelatedInstances(applRel,iePattern);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_RELATION](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.InstanceElementTest.testCheckRelLccMeq()
 test.highqsoft.avalon.InstanceElementTest.testInstanceRelatedAgainstInverseRelated()
 test.highqsoft.avalon.InstanceElementTest.testInstancesRelationNames()
 test.highqsoft.avalon.InstanceElementTest.testInstancesRelations()
 test.highqsoft.avalon.MultiSessionTest.threadCompareValues.run()
 test.highqsoft.avalon.MultiSessionTest.threadInstancesRelations.run()
 test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationNames.run()
 test.highqsoft.avalon.ValueMatrixTest.testCompareColumns()
 test.highqsoft.avalon.ValueMatrixTest.testCompareValues()
 test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromMeasSm()

4.16.13 `getRelatedInstancesByRelationship` of interface `InstanceElement`

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.

Return:

[InstanceElementIterator](#) ieIterator *The related instances.*

Parameter:

[Relationship](#) ieRelationship (in) *The requested relationship.*

[Pattern](#) iePattern (in) *The name or the search pattern for the related instance names.*

Java Calling Sequence:

```
InstanceElementIterator ieIterator = instElem.getRelatedInstancesByRelationship(ieRelationship,iePattern);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_RELATIONSHIP](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.InstanceElementTest.testInstancesRelationChild()
 test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationChild.run()



```
test.highqsoft.odsapi.read.InstanceRelationTest.testReadRelatedInstances()
test.highqsoft.odsapi.CreateMeasurementTest.testDeleteValues()
test.highqsoft.odsapi.WriteInstanceElementTest.testCopyTest()
```

4.16.14 getRights of interface InstanceElement

Purpose:

Retrieve access control list information of the given object.

Return:

[ACLSequence](#) aclEntries *The access control list entries of the given application element.*

Parameter:

None.

Java Calling Sequence:

```
ACL[] aclEntries = instElem.getRights();
```

Java Example:

```
// Get the application structure.
ApplicationStructure as = aoSession.getApplicationStructure();

// Get ApplElemAccess Interface
ApplElemAccess aea = aoSession.getApplElemAccess();

// Get the all application elements.
ApplicationElement aeList[] = as.getElements("*");

for (int aeIndex = 0; aeIndex < aeList.length; aeIndex++) {
    ACL [] aclList;           // The access control list.
    int secLevel;             // security level
    // Name of the application element
    String aeName = aeList[aeIndex].getName();

    // Get the security level
    secLevel = aeList[aeIndex].getSecurityLevel();

    if ((secLevel & SecurityLevel.INSTANCE_SECURITY) != 0) {
        // Get the instances of the application element
        InstanceElement ieObj;
        InstanceElementIterator iei = aeList[aeIndex].getInstances("*");
        if (iei != null) {
            do {
                ieObj = iei.nextOne();
                if (ieObj != null) {
                    // Id of the instance,
                    T_LONGLONG iid = ieObj.getId();
                    // Get the access control list.
                    aclList = ieObj.getRights();
                    for (int aclIndex = 0;
                        aclIndex < aclList.length;
                        aclIndex++) {
                        System.out.println(aeName + ", " + iid.low + ", " +
                            aclList[aclIndex].usergroupId.low + ", " +
                            aclList[aclIndex].rights);
                    }
                    // Destroy the instance element, the object is not
                    // used anymore at the client.
                    ieObj.destroy();
                }
            } while (iei != null);
        }
    }
}
```

```

        // Destroy the instance iterator, the object is not
        // used anymore at the client.
        iei.destroy();
    }
} // Only when security level is INSTANCE_ELEMENT
} // for all application elements.

```

See also:

[getInstanceRights](#) of interface [ApplElemAccess](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.SecurityTest.testRights()
test.highqsoft.odsapi.WriteInstanceElementTest.testReadInstancesWithRights()

```

4.16.15 getValue of interface InstanceElement**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](#).

Return:

[NameValueUnit](#) nameValueUnit *The attribute value.*

Parameter:

[Name](#) attrName (in) *The name of the requested attribute.*

Java Calling Sequence:

```
NameValueUnit nameValueUnit = instElem.getValue(attrName);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCheckRename()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemAltUnit()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemModAttr()
test.highqsoft.odsapi.read.AoFactoryTest.testNewFactoryUser()
test.highqsoft.odsapi.read.AttributeValueTest.testBlobAttributeValues()
test.highqsoft.odsapi.read.ExtendedQueryTest.testAllSeqAttr()

```



```

test.highqsoft.odsapi.read.UnitTest.testSetValueInUnit()
test.highqsoft.odsapi.CreateInstanceTestCase.newInstance()
test.highqsoft.odsapi.TestConnectServer.testConnectServer()
test.highqsoft.odsapi.UpdateInstanceElementTest.testGenerationParameters()
....

```

4.16.16 getValueByBaseName of interface InstanceElement

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.

Return:

[NameValueUnit](#) nameValueUnit *The attribute value.*

Parameter:

[Name](#) baseAttrName (in) *The base name of the requested attribute.*

Java Calling Sequence:

```
NameValueUnit nameValueUnit = instElem.getValueByBaseName(baseAttrName);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.MeasurementTest.testManyMeasurement()
test.highqsoft.avalon.MultiSessionTest.threadCompareValues.run()
test.highqsoft.avalon.ValueMatrixTest.testCompareColumns()
test.highqsoft.avalon.ValueMatrixTest.testCompareValues()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromSmSelectMea()
test.highqsoft.avalon.ValueMatrixTest.testUnitConversion()
test.highqsoft.avalon.ValueMatrixTest.testValueMatrixGetValueVector()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElem()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemAltUnit()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemModAttr()
....

```

4.16.17 getValueInUnit of interface InstanceElement

Purpose:

Get the attribute value (name, value and unit) of the given attribute of the instance element.

Return:

[NameValueUnit](#) nameValueUnit *The attribute value, value converted to the requested unit.*

Parameter:

[NameUnit](#) attr (in) *The name of the requested attribute and the unit of the attribute value.*

Java Calling Sequence:

```
NameValueUnit nameValueUnit = instElem.getValueInUnit(attr);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_INCOMPATIBLE_UNITS](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.read.UnitTest.testGetValueInUnit()
```

4.16.18 getValueSeq of interface InstanceElement**Purpose:**

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.

Return:

[NameValueUnitSequence](#) values *The sequence of the attribute values.*

Parameter:

[NameSequence](#) attrNames (in) *The names of the attributes to be reported.*

Java Calling Sequence:

```
Name attrNames[];  
...  
NameValueUnit values[] = ieObj.getValueSeq(attrNames);
```

Since:

ASAM ODS 5.0

See also:

[getValue](#) of interface [InstanceElement](#)
[setValue](#) of interface [InstanceElement](#)
[setValueSeq](#) of interface [InstanceElement](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.16.19 listAttributes of interface InstanceElement

Purpose:

Get the attribute names from the instance element. The attributes reserved for a relation are not listed.

Return:

[NameSequence](#) ieNames *The names of the attributes.*

Parameter:

[Pattern](#) iaPattern (in) *The name or the search pattern for the attribute names.*

[AttrType](#) aType (in) *The requested attribute type.*

Java Calling Sequence:

```
AttrType aType = AttrType.ALL;
Name[] ieNames = instElem.listAttributes(iaPattern, aType);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_ATTRIBUTE_TYPE](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.InstanceAttributeTest.testCheckInstanceAttribute()
test.highqsoft.odsapi.WriteInstanceElementTest.testUpdateInstanceWithoutId()
```

4.16.20 listRelatedInstances of interface InstanceElement

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.

Return:

[NameIterator](#) ieNameIterator *The names of the related instances.*

Parameter:

[ApplicationRelation](#) ieRelation (in) *The application relation.*

[Pattern](#) iePattern (in) *The name or the search pattern for the related instance names.*

Java Calling Sequence:

```
NameIterator ieNameIterator = instElem.listRelatedInstances(ieRelation, iePattern);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)

[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.InstanceElementTest.testInstancesRelationNames()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationNames.run()
test.highqsoft.odsapi.read.InstanceRelationTest.testGetSmRelatedInstances()

```

4.16.21 listRelatedInstancesByRelationship of interface InstanceElement

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.

Return:

[NameIterator](#) ieNameIterator *The names of the related instances.*

Parameter:

[Relationship](#) ieRelationship (in) *The requested relationship.*

[Pattern](#) iePattern (in) *The name or the search pattern for the related instance names.*

Java Calling Sequence:

```
NameIterator ieNameIterator = instElem.listRelatedInstancesByRelationship(ieRelationship,iePattern);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_RELATIONSHIP](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.InstanceElementTest.testInstancesRelationChild()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationChild.run()

```

4.16.22 removeInstanceAttribute of interface InstanceElement

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.

Return:

void



Parameter:

[Name](#) attrName (in) *The name of the attribute to be removed.*

Java Calling Sequence:

```
instElem.removeInstanceAttribute(attrName);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_FOUND](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.16.23 removeRelation of interface InstanceElement

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.

Return:

void

Parameter:

[ApplicationRelation](#) applRel (in) *The relation to be removed.*

[InstanceElement](#) instElem_nm (in) *The instance element for specific delete from n:m relations.*

Java Calling Sequence:

```
instElem.removeRelation(applRel, instElem_nm);
```

Errors:

[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](#)

Tested by:

test.highqsoft.odsapi.DeleteInstanceElementTest.testDeleteInstancesRec()

4.16.24 renameInstanceAttribute of interface InstanceElement

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.

Return:

void

Parameter:

[Name](#) oldName (in) *The old instance attribute name.*

[Name](#) newName (in) *The new instance attribute name.*

Java Calling Sequence:

```
instElem.renameInstanceAttribute(oldName,newName);
```

Errors:

[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](#)

4.16.25 setInitialRights of interface InstanceElement

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.

Return:

void

Parameter:

[InstanceElement](#) usergroup (in) *The usergroup for which the initial rights will be modified.*

[T_LONG](#) rights (in) *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.*

[T_LONGLONG](#) refAid (in) *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.*

[RightsSet](#) set (in) *What to do with the new initial rights.*

Java Calling Sequence:

```
instElem.setInitialRights(usergroup, rights, refAid, set);
```

See also:

[setInstanceInitialRights](#) of interface [ApplElemAccess](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.security.InstanceCheckTest.testSetInstanceInitialInstRights()  
test.highqsoft.odsapi.security.InstanceCheckTest.testSetInstanceInitialRights()  
test.highqsoft.odsapi.security.SecurityTestCase.setInstanceInitialRights()  
test.highqsoft.odsapi.CreateInstanceElementTest.testCreateElementIniRights()
```

4.16.26 setName of interface InstanceElement

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.

Return:

void

Parameter:

[Name](#) iaName (in) *The instance attribute name.*

Java Calling Sequence:

```
instElem.setName(iaName);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.UpdateInstanceElementTest.testSetName()  
test.highqsoft.odsapi.WriteInstanceElementTest.testLongName()
```

4.16.27 setRights of interface InstanceElement

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.

Return:

void

Parameter:

[InstanceElement](#) usergroup (in) *The usergroup for which the rights will be modified.*

[T_LONG](#) rights (in) *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.*

[RightsSet](#) set (in) *What to do with the new right.*

Java Calling Sequence:

```
instElem.setRights(usergroup, rights, set);
```

See also:

[setInstanceRights](#) of interface [ApplElemAccess](#)

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.security.InstanceCheckTest.testSetInstanceInitialInstRights()
test.highqsoft.odsapi.security.InstanceCheckTest.testSetInstanceInstRights()
test.highqsoft.odsapi.security.InstanceCheckTest.testSubTestRights()
test.highqsoft.odsapi.security.SecurityTestCase.setInstanceRights()
```

4.16.28 setValue of interface InstanceElement

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 value of a relation attribute, use the method createRelation.



Note:

See [Transaction handling in ODS API](#).(p. 14)

Return:

void

Parameter:

[NameValueUnit](#) value (in) *The value to be set in the instance element.*

Java Calling Sequence:

```
instElem.setValue(value);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_DUPLICATE_VALUE](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemAltUnit()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemModAttr()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemModLength()
test.highqsoft.odsapi.changemodel.ApplElemCreateTest.testCreateInstElemModObligatory()
test.highqsoft.odsapi.read.AoFactoryTest.testNewFactoryUser()
test.highqsoft.odsapi.read.AsamPathTest.testCreateAsamPathSpaces()
test.highqsoft.odsapi.read.ExtendedQueryTest.testCreateTestDataOuterJoin()
test.highqsoft.odsapi.read.UnitTest.testSetValueInUnit()
test.highqsoft.odsapi.security.InstanceCheckTest.testSubTestRights()
test.highqsoft.odsapi.security.InstanceCheckTest.testSuperuserFlag()
....
```

4.16.29 setValueSeq of interface InstanceElement

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.

Note:

See [Transaction handling in ODS API](#).(p. 14)

Return:

void

Parameter:

[NameValueUnitSequence](#) values (in) *The sequence of the values to be set at the instance element.*

Java Calling Sequence:

```
instElem.setValueSeq(values);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_DUPLICATE_VALUE](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.security.InstanceCheckTest.testSubTestRights()
test.highqsoft.odsapi.UpdateInstanceElementTest.testGenerationParameters()
test.highqsoft.odsapi.WriteInstanceElementTest.testWrongAttributeName()
```

4.16.30 shallowCopy of interface InstanceElement**Purpose:**

Provides an easy-to-use and effective copy mechanism for instance elements inside the server. The new instance element 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.

Return:

[InstanceElement](#) newInstElem *The reference to the copied instance element.*

Parameter:

[T_STRING](#) newName (in) *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.*

[T_STRING](#) newVersion (in) *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);
```

Since:

ASAM ODS 4.1

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.WriteInstanceElementTest.testCopyTest()
```



4.16.31 upcastMeasurement of interface InstanceElement

Purpose:

Cast an instance element to a measurement. There are some object-oriented languages which do not allow this cast.

Return:

[Measurement](#) measurement *The instance of type measurement.*

Parameter:

None.

Java Calling Sequence:

```
Measurement measurement = instElem.upcastMeasurement();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_INVALID_BASETYPE](#)

Tested by:

```
test.highqsoft.avalon.MultiSessionTest.threadCreateValueMatrix.run()
test.highqsoft.avalon.ValueMatrixTest.testCheckUnit()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrix()
test.highqsoft.avalon.ValueMatrixTest.testUnitConversion()
test.highqsoft.odsapi.read.MeasurementTest.testValueMatrixIndep()
test.highqsoft.odsapi.read.MeasurementTest.testValueMatrixUnit()
test.highqsoft.odsapi.read.UnitTest.testValuematrixCheck()
test.highqsoft.odsapi.read.UnitTest.testValuematrixGet()
test.highqsoft.odsapi.CreateInstanceTestCase.checkMeasurementInstances()
test.highqsoft.odsapi.UpdateMeasurementTest.testCheckMeasurement()
....
```

4.16.32 upcastSubMatrix of interface InstanceElement

Purpose:

Cast an instance element to a submatrix. There are some object-oriented languages which do not allow this cast.

Return:

[SubMatrix](#) subMatrix *The instance of type submatrix.*

Parameter:

None.

Java Calling Sequence:

```
SubMatrix subMatrix = instElem.upcastSubMatrix();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_INVALID_BASETYPE](#)

Tested by:

test.highqsoft.avalon.MultiSessionTest.threadCompareValues.run()
 test.highqsoft.avalon.MultiSessionTest.threadCreateValueMatrixFromSm.run()
 test.highqsoft.avalon.ValueMatrixTest.testCompareColumns()
 test.highqsoft.avalon.ValueMatrixTest.testCompareValues()
 test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromMeaSm()
 test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromSm()
 test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromSmSelectMea()
 test.highqsoft.avalon.ValueMatrixTest.testListColumns()
 test.highqsoft.avalon.ValueMatrixTest.testSubMatrixListColumns()
 test.highqsoft.avalon.ValueMatrixTest.testValueMatrixGetMeaPoint()

4.17 InstanceElementIterator

Package:

org.asam.ods

Description:

The ASAM ODS instance element iterator interface.

4.17.1 destroy of interface InstanceElementIterator

Purpose:

Destroy the iterator and free the associated memory.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
instElemIterator.destroy();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

org.asam.ods.InstanceElementIteratorPOA._invoke()




```

test.highqsoft.avalon.InstanceElementTest.testCheckRelLccMeq()
test.highqsoft.avalon.InstanceElementTest.testCorbaReferences()
test.highqsoft.avalon.InstanceElementTest.testInstanceRelatedAgainstInverseRelated()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationChild()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationNames()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelations()
test.highqsoft.avalon.MeasurementTest.testGetLoclColumnInstances()
test.highqsoft.avalon.MeasurementTest.testMeasurement()
test.highqsoft.avalon.MeasurementTest.testMeasurementCompareValues()
....

```

4.17.2 getCount of interface InstanceElementIterator

Purpose:

Get the total number of elements accessible by the iterator.

Return:

[T_LONG](#) instanceCount *The number of elements accessible by the iterator.*

Parameter:

None.

Java Calling Sequence:

```
T_LONG instanceCount = instElemIterator.getCount();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.InstanceElementTest.testCheckRelLccMeq()
test.highqsoft.avalon.InstanceElementTest.testInstanceRelatedAgainstInverseRelated()
test.highqsoft.avalon.InstanceElementTest.testInstancesNames()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationChild()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationNames()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelations()
test.highqsoft.avalon.MeasurementTest.testGetLoclColumnInstances()
test.highqsoft.avalon.MeasurementTest.testMeasurement()
test.highqsoft.avalon.MeasurementTest.testMeasurementCompareValues()
test.highqsoft.avalon.MeasurementTest.testMeasurementCompareValuesExt()
....

```

4.17.3 nextN of interface InstanceElementIterator

Purpose:

Get the next n elements from the sequence.

Return:

[InstanceElementSequence](#) instElems *The next n instance elements from the instance sequence.*

Parameter:

[T_LONG](#) how_many (in) *The number of requested elements.*

Java Calling Sequence:

```
InstanceElement[] instElems = instElemIterator.nextN(how_many);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.odsapi.read.AsamPathTest.testCreateAsamPathSpaces()
test.highqsoft.odsapi.CreateInstanceElementTest.testAeCreateInstances()
test.highqsoft.odsapi.CreateInstanceTestCase.createMeqQtyRelationApplStruct()
test.highqsoft.odsapi.CreateMeasurementTest.testCreateMeasurementCommit()
test.highqsoft.odsapi.CreateMeasurementTest.testCreateMeasurementDoubleRel()
test.highqsoft.odsapi.UpdateMeasurementTest.createMeasurementBody()
test.highqsoft.odsapi.WriteInstanceElementTest.testCreateInstanceRelations()
```

4.17.4 nextOne of interface InstanceElementIterator**Purpose:**

Get the next element from the sequence.

Return:

[InstanceElement](#) instElem *The next instance element from the instance sequence.*

Parameter:

None.

Java Calling Sequence:

```
InstanceElement instElem = instElemIterator.nextOne();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.InstanceElementTest.testCorbaReferences()
test.highqsoft.avalon.InstanceElementTest.testInstanceRelatedAgainstInverseRelated()
test.highqsoft.avalon.InstanceElementTest.testInstancesNames()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationChild()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationNames()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelations()
test.highqsoft.avalon.MeasurementTest.testMeasurement()
```



```
test.highqsoft.avalon.MeasurementTest.testMeasurementCompareValues()
test.highqsoft.avalon.MeasurementTest.testMeasurementCompareValuesExt()
test.highqsoft.avalon.MultiSessionTest.threadCompareValues.run()
....
```

4.17.5 reset of interface InstanceElementIterator

Purpose:

Reset the pointer in the element sequence to the first element.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
instElemIterator.reset();
```

Errors:

AO_CONNECTION_LOST
 AO_IMPLEMENTATION_PROBLEM
 AO_NOT_IMPLEMENTED
 AO_NO_MEMORY
 AO_SESSION_NOT_ACTIVE

Tested by:

```
test.highqsoft.avalon.InstanceElementTest.testInstanceRelatedAgainstInverseRelated()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelations()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelations.run()
test.highqsoft.odsapi.read.UnitTest.testGetValueInUnit()
test.highqsoft.odsapi.security.ApplModelCheckTest.testSecurityLevel()
test.highqsoft.odsapi.security.InstanceCheckTest.testSetInstanceInitialInstRights()
test.highqsoft.odsapi.security.InstanceCheckTest.testSetInstanceInitialInstRights()
test.highqsoft.odsapi.security.InstanceCheckTest.testSetInstanceInitialRights()
test.highqsoft.odsapi.security.InstanceCheckTest.testSetInstanceInstRights()
test.highqsoft.odsapi.security.SecurityTestCase.setInstanceInitialRights()
....
```

4.18 Measurement

Package:

org.asam.ods

Description:

The ASAM ODS measurement interface.

Derived:

[InstanceElement](#)

4.18.1 createSMatLink of interface Measurement

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.

Note:

Not yet implemented, SMatLink are not well defined in ASAM ODS.

Return:

[SMatLink](#) smLink *The new submatrix link.*

Parameter:

None.

Java Calling Sequence:

```
SMatLink smLink = measurement.createSMatLink();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.18.2 getSMatLinks of interface Measurement

Purpose:

Get the list of the submatrix links .

Note:

Not yet implemented, SMatLink are not well defined in ASAM ODS.

Return:

[SMatLinkSequence](#) smLinks *The available submatrix links.*

Parameter:

None.

Java Calling Sequence:

```
SMatLink[] smLinks = measurement.getSMatLinks();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.18.3 getValueMatrix of interface Measurement

Purpose:

Get the value matrix of a measurement.

Return:

[ValueMatrix](#) vm *The value matrix.*

Parameter:

None.

Java Calling Sequence:

```
ValueMatrix vm = measurement.getValueMatrix();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.MultiSessionTest.threadCreateValueMatrix.run()
test.highqsoft.avalon.ValueMatrixTest.testCheckUnit()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrix()
test.highqsoft.avalon.ValueMatrixTest.testUnitConversion()
test.highqsoft.odsapi.read.MeasurementTest.testValueMatrixIndep()
test.highqsoft.odsapi.read.MeasurementTest.testValueMatrixUnit()
test.highqsoft.odsapi.read.UnitTest.testValuematrixCheck()
test.highqsoft.odsapi.CreateInstanceTestCase.checkMeasurementInstances()
test.highqsoft.odsapi.UpdateMeasurementTest.testCheckMeasurement()
test.highqsoft.odsapi.UpdateMeasurementTest.ThreadViewValues.run()
```

4.18.4 removeSMatLink of interface Measurement

Purpose:

Remove a submatrix link.

Note:

Not yet implemented, SMatLink are not well defined in ASAM ODS.

Return:

void

Parameter:

[SMatLink](#) smLink (in) *The submatrix link to be removed.*

Java Calling Sequence:

```
measurement.removeSMatLink(smLink);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)

[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_FOUND](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.19 NameIterator

Package:

org.asam.ods

Description:

The ASAM ODS name iterator interface.

The NameIterator returns at the `nextOne()` method a String. The String is not defined as a corba interface but as an corba type. Corba does not support the transport of a NULL-value. When after the last element in the iterator a request is done, an Corba exception will occure.

4.19.1 destroy of interface NameIterator

Purpose:

Destroy the iterator and free the associated memory.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
nameIterator.destroy();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```

org.asam.ods.NameIteratorPOA._invoke()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationChild()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationNames()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationChild.run()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationNames.run()

```

4.19.2 getCount of interface NameIterator

Purpose:

Get the total number of elements accessible by the iterator.



Return:

[T_LONG](#) nameCount *The number of elements accessible by the iterator.*

Parameter:

None.

Java Calling Sequence:

```
T_LONG nameCount = nameIterator.getCount();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.AoSessionTest.testAoSessionContextRead()
test.highqsoft.avalon.InstanceElementTest.testInstancesNames()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationChild()
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationNames()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationChild.run()
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationNames.run()
test.highqsoft.odsapi.read.InstanceRelationTest.testGetSmRelatedInstances()
test.highqsoft.odsapi.DeleteInstanceElementTest.testApplElemAccessDeleteInstances()
test.highqsoft.odsapi.XATFCopyTest.testCompareSecurityInformation()
```

4.19.3 nextN of interface NameIterator

Purpose:

Get the next n elements from the sequence.

Return:

[NameSequence](#) names *The next n names from the name sequence.*

Parameter:

[T_LONG](#) how_many (in) *The number of requested elements.*

Java Calling Sequence:

```
Name[] names = nameIterator.nextN(how_many);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.InstanceElementTest.testInstancesNames()
```

4.19.4 nextOne of interface NameIterator

Purpose:

Get the next element from the sequence.

Return:

[Name](#) name *The next name from the name sequence.*

Parameter:

None.

Java Calling Sequence:

```
Name name = nameIterator.nextOne();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.AoSessionTest.testAoSessionContextRead()  
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationChild()  
test.highqsoft.avalon.InstanceElementTest.testInstancesRelationNames()  
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationChild.run()  
test.highqsoft.avalon.MultiSessionTest.threadInstancesRelationNames.run()  
test.highqsoft.odsapi.DeleteInstanceElementTest.testApplElemAccessDeleteInstances()  
test.highqsoft.odsapi.XATFCopyTest.testCompareSecurityInformation()
```

4.19.5 reset of interface NameIterator

Purpose:

Reset the pointer in the element sequence to the first element.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
nameIterator.reset();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)



4.20 NameValueIterator

Package:

org.asam.ods

Description:

The ASAM ODS name-value iterator interface.

The NameValueIterator returns at the `nextOne()` method a NameValue. The NameValue is not defined as a corba interface but as an corba structure. Corba does not support the transport of a NULL-Structure. When after the last element in the iterator a request is done, an Corba exception will occure.

4.20.1 destroy of interface NameValueIterator

Purpose:

Destroy the iterator and free the associated memory.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
nameValueIterator.destroy();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
org.asam.ods.NameValueIteratorPOA._invoke()  
test.highqsoft.avalon.AoSessionTest.testAoSessionContextRead()
```

4.20.2 getCount of interface NameValueIterator

Purpose:

Get the total number of elements accessible by the iterator.

Return:

[T_LONG](#) nvCount *The number of elements accessible by the iterator.*

Parameter:

None.

Java Calling Sequence:

```
T_LONG nvCount = nameValueIterator.getCount();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.AoSessionTest.testAoSessionContextRead()
test.highqsoft.avalon.IteratorTest.testNameValueIteratorReset()

4.20.3 nextN of interface NameValueIterator**Purpose:**

Get the next n elements from the sequence.

Return:

[NameValueSequence](#) nameValues *The next n name-value pairs from the name-value pair sequence.*

Parameter:

[T_LONG](#) how_many (in) *The number of requested elements.*

Java Calling Sequence:

```
NameValue[] nameValues = nameValueIterator.nextN(how_many);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.AoSessionTest.testAoSessionContextRead()

4.20.4 nextOne of interface NameValueIterator**Purpose:**

Get the next element from the sequence.

Return:

[NameValue](#) nameValue *The next name-value pair from the name-value pair sequence.*

Parameter:

None.

Java Calling Sequence:

```
NameValue nameValue = nameValueIterator.nextOne();
```



Errors:

AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

test.highqsoft.avalon.AoSessionTest.testAoSessionContextRead()
test.highqsoft.avalon.IteratorTest.testNameValueIteratorReset()

4.20.5 reset of interface NameValueIterator

Purpose:

Reset the pointer in the element sequence to the first element.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
nameValueIterator.reset();
```

Errors:

AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

test.highqsoft.avalon.AoSessionTest.testAoSessionContextRead()

4.21 NameValueUnitIdIterator

Package:

org.asam.ods

Description:

The ASAM ODS name-value-unitId iterator interface. This interface is identical with the NameValueUnitIterator, except the unit is given as an Id instead of a string.

The NameValueUnitIdIterator returns at the `nextOne()` method a NameValueUnitId. The NameValueUnitId is not defined as a corba interface but as an corba structure. Corba does not support the transport of a NULL-Structure. When after the last element in the iterator a request is done, an Corba exception will occur.

4.21.1 destroy of interface NameValueUnitIdIterator

Purpose:

Destroy the iterator and free the associated memory.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
nameValueUnitIdIterator.destroy();
```

Errors:

AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

```
org.asam.ods.NameValueUnitIdIteratorPOA._invoke()
```

4.21.2 getCount of interface NameValueUnitIdIterator

Purpose:

Get the total number of elements accessible by the iterator.

Return:

[T_LONG](#) nameValueUnitIdCount *The number of elements accessible by the iterator.*

Parameter:

None.

Java Calling Sequence:

```
T_LONG nvuIdCount = nameValueUnitIdIterator.getCount();
```

Errors:

AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

4.21.3 nextN of interface NameValueUnitIdIterator

Purpose:

Get the next n elements from the sequence.



Return:

[NameValueSeqUnitId](#) nameValueUnitIdSeq *The next n name-value-unit tuples from the name-value-unit tuple sequence.*

Parameter:

[T_LONG](#) how_many (in) *The number of requested elements.*

Java Calling Sequence:

```
NameValueUnitId[] nameValueUnitIds = nameValueUnitIdIterator.nextN(how_many);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.21.4 nextOne of interface NameValueUnitIdIterator**Purpose:**

Get the next element from the sequence.

Return:

[NameValueUnitId](#) nameValueUnitId *The next name-value-unitId tuple from the name-value-unitId tuple sequence.*

Parameter:

None.

Java Calling Sequence:

```
NameValueUnitId nameValueUnitId = nameValueUnitIdIterator.nextOne();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.21.5 reset of interface NameValueUnitIdIterator**Purpose:**

Reset the pointer in the element sequence to the first element.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
nameValueUnitIdIterator.reset();
```

Errors:

```
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE
```

4.22 NameValueUnitIterator

Package:

```
org.asam.ods
```

Description:

The ASAM ODS name-value-unit iterator interface. This interface is identical with the NameValueUnitIdIterator, except the unit is given as a string instead of an Id.

The NameValueUnitIterator returns at the `nextOne()` method a NameValueUnit. The NameValueUnit is not defined as a corba interface but as an corba structure. Corba does not support the transport of a NULL-Structure. When after the last element in the iterator a request is done, an Corba exception will occur. A Stackprint is show below:

Caused an ERROR

-----BEGIN server-side stack trace-----

```
org.omg.CORBA.UNKNOWN: vmcid: SUN minor code: 202 completed: Maybe
at com.sun.corba.se.impl.logging.ORBUtilSystemException.runtimeException(ORBUtilSystemException.java:8365)
at com.sun.corba.se.impl.protocol.CorbaMessageMediatorImpl.convertThrowableToSystemException(CorbaMessageMediatorImpl.java:258)
at com.sun.corba.se.impl.protocol.CorbaMessageMediatorImpl.handleThrowableDuringServerDispatch(CorbaMessageMediatorImpl.java:1680)
at com.sun.corba.se.impl.protocol.CorbaMessageMediatorImpl.handleRequestRequest(CorbaMessageMediatorImpl.java:1540)
at com.sun.corba.se.impl.protocol.CorbaMessageMediatorImpl.handleRequest(CorbaMessageMediatorImpl.java:922)
at com.sun.corba.se.impl.protocol.giopmsgheaders.RequestMessage_1_2.callback(RequestMessage_1_2.java:181)
at com.sun.corba.se.impl.protocol.CorbaMessageMediatorImpl.handleRequest(CorbaMessageMediatorImpl.java:694)
at com.sun.corba.se.impl.transport.SocketOrChannelConnectionImpl.dispatch(SocketOrChannelConnectionImpl.java:451)
at com.sun.corba.se.impl.transport.SocketOrChannelConnectionImpl.doWork(SocketOrChannelConnectionImpl.java:1187)
at com.sun.corba.se.impl.orbutil.threadpool.ThreadPoolImpl$WorkerThread.run(ThreadPoolImpl.java:417)
Caused by: java.lang.NullPointerException
at org.asam.ods.NameValueUnitHelper.write(NameValueUnitHelper.java:87)
at org.asam.ods.NameValueUnitIteratorPOA._invoke(NameValueUnitIteratorPOA.java:150)
at com.sun.corba.se.impl.protocol.CorbaServerRequestDispatcherImpl.dispatchToServant(CorbaServerRequestDispatcherImpl.java:189)
at com.sun.corba.se.impl.protocol.CorbaServerRequestDispatcherImpl.dispatch(CorbaServerRequestDispatcherImpl.java:189)
... 8 more
```

-----END server-side stack trace-----

```
org.omg.CORBA.UNKNOWN: -----BEGIN server-side stack trace-----
org.omg.CORBA.UNKNOWN: vmcid: SUN minor code: 202 completed: Maybe
at com.sun.corba.se.impl.logging.ORBUtilSystemException.runtimeException(ORBUtilSystemException.java:8365)
at com.sun.corba.se.impl.protocol.CorbaMessageMediatorImpl.convertThrowableToSystemException(CorbaMessageMediatorImpl.java:258)
at com.sun.corba.se.impl.protocol.CorbaMessageMediatorImpl.handleThrowableDuringServerDispatch(CorbaMessageMediatorImpl.java:1680)
at com.sun.corba.se.impl.protocol.CorbaMessageMediatorImpl.handleRequestRequest(CorbaMessageMediatorImpl.java:1540)
at com.sun.corba.se.impl.protocol.CorbaMessageMediatorImpl.handleRequest(CorbaMessageMediatorImpl.java:922)
at com.sun.corba.se.impl.protocol.giopmsgheaders.RequestMessage_1_2.callback(RequestMessage_1_2.java:181)
at com.sun.corba.se.impl.protocol.CorbaMessageMediatorImpl.handleRequest(CorbaMessageMediatorImpl.java:694)
```



```

at com.sun.corba.se.impl.transport.SocketOrChannelConnectionImpl.dispatch(SocketOrChannelConnectionImpl.java:451)
at com.sun.corba.se.impl.transport.SocketOrChannelConnectionImpl.doWork(SocketOrChannelConnectionImpl.java:1187)
at com.sun.corba.se.impl.orbutil.threadpool.ThreadPoolImpl$WorkerThread.run(ThreadPoolImpl.java:417)
Caused by: java.lang.NullPointerException
at org.asam.ods.NameValueUnitHelper.write(NameValueUnitHelper.java:87)
at org.asam.ods.NameValueUnitIteratorPOA._invoke(NameValueUnitIteratorPOA.java:150)
at com.sun.corba.se.impl.protocol.CorbaServerRequestDispatcherImpl.dispatchToServant(CorbaServerRequestDispatcherImpl.java:189)
at com.sun.corba.se.impl.protocol.CorbaServerRequestDispatcherImpl.dispatch(CorbaServerRequestDispatcherImpl.java:189)
... 8 more

-----END server-side stack trace----- vmcid: SUN minor code: 202 completed: Maybe
at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:39)
at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:27)
at java.lang.reflect.Constructor.newInstance(Constructor.java:494)
at com.sun.corba.se.impl.protocol.giopmsgheaders.MessageBase.getSystemException(MessageBase.java:902)
at com.sun.corba.se.impl.protocol.giopmsgheaders.ReplyMessage_1_2.getSystemException(ReplyMessage_1_2.java:99)
at com.sun.corba.se.impl.protocol.CorbaMessageMediatorImpl.getSystemExceptionReply(CorbaMessageMediatorImpl.java:572)
at com.sun.corba.se.impl.protocol.CorbaClientRequestDispatcherImpl.processResponse(CorbaClientRequestDispatcherImpl.java:129)
at com.sun.corba.se.impl.protocol.CorbaClientRequestDispatcherImpl.marshalingComplete(CorbaClientRequestDispatcherImpl.java:129)
at com.sun.corba.se.impl.protocol.CorbaClientDelegateImpl.invoke(CorbaClientDelegateImpl.java:129)
at org.omg.CORBA.portable.ObjectImpl._invoke(ObjectImpl.java:457)
at org.asam.ods._NameValueUnitIteratorStub.nextOne(_NameValueUnitIteratorStub.java:150)
at test.highqsoft.avalon.ValueMatrixTest.testValueMatrixGetMeaPoint(ValueMatrixTest.java:1545)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)

```

Use the following code to read all elements from the iterator without an corba exception:

```

NameValueUnitIterator nvuIter = vmObj.getValueMeaPoint(0);
if (nvuIter != null) {
    NameValueUnit nvu;
    int noPoints = nvuIter.getCount();
    for (int i = 0; i < noPoints; i++) {
        nvu = nvuIter.nextOne();
        //
        // TODO
        //
    }
    nvuIter.destroy();
}

```

4.22.1 destroy of interface NameValueUnitIterator

Purpose:

Destroy the iterator and free the associated memory.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
nameValueUnitIterator.destroy();
```

Errors:

[AO_CONNECTION_LOST](#)

[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

org.asam.ods.NameValueUnitIteratorPOA._invoke()

4.22.2 getCount of interface NameValueUnitIterator**Purpose:**

Get the total number of elements accessible by the iterator.

Return:

[T_LONG](#) nvuCount *The number of elements accessible by the iterator.*

Parameter:

None.

Java Calling Sequence:

```
T_LONG nvuCount = nameValueUnitIterator.getCount();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.ValueMatrixTest.testValueMatrixGetMeaPoint()

4.22.3 nextN of interface NameValueUnitIterator**Purpose:**

Get the next n elements from the sequence.

Return:

[NameValueUnitSequence](#) nameValueUnits *The next n name-value-unit tuples from the name-value-unit tuple sequence.*

Parameter:

[T_LONG](#) how_many (in) *The number of requested elements.*

Java Calling Sequence:

```
NameValueUnit[] nameValueUnits = nameValueUnitIterator.nextN(how_many);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)



Tested by:

test.highqsoft.avalon.ValueMatrixTest.testValueMatrixGetMeaPoint()

4.22.4 nextOne of interface NameValueUnitIterator**Purpose:**

Get the next element from the sequence.

Return:

[NameValueUnit](#) nameValueUnit *The next name-value-unit tuple from the name-value-unit tuple sequence.*

Parameter:

None.

Java Calling Sequence:

```
NameValueUnit nameValueUnit = nameValueUnitIterator.nextOne();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.ValueMatrixTest.testValueMatrixGetMeaPoint()

4.22.5 reset of interface NameValueUnitIterator**Purpose:**

Reset the pointer in the element sequence to the first element.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
nameValueUnitIterator.reset();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.ValueMatrixTest.testValueMatrixGetMeaPoint()

4.23 NameValueUnitSequenceIterator

Package:

org.asam.ods

Description:

The name value unit sequence iterator. The table iterator as query result.

4.23.1 destroy of interface NameValueUnitSequenceIterator

Purpose:

Destroy the iterator and free the associated memory.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
nameValueUnitSequenceIterator.destroy();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

org.asam.ods.NameValueUnitSequenceIteratorPOA._invoke()

4.23.2 getCount of interface NameValueUnitSequenceIterator

Purpose:

Get the total number of elements accessible by the iterator.

Return:

[T_LONG](#) nameValueUnitCount *The number of elements accessible by the iterator.*

Parameter:

None.

Java Calling Sequence:

```
T_LONG nameValueUnitCount = nameValueUnitSequenceIterator.getCount();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.23.3 nextN of interface NameValueUnitSequenceIterator

Purpose:

Get the next n elements from the sequence.

Return:

[NameValueSeqUnitSequence](#) nameValueSeqUnits *The next n values of the name-value-unit sequence. For each NameValueUnit the next n values are stored in the value sequence.*

Parameter:

[T_LONG](#) how_many (in) *The number of requested elements.*

Java Calling Sequence:

```
NameValueSeqUnit[] nameValueSeqUnits = nameValueUnitSequenceIterator.nextN(how_many);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.23.4 nextOne of interface NameValueUnitSequenceIterator

Purpose:

Get the next element from the iterator.

Return:

[NameValueSeqUnit](#) nameValueSeqUnit *The next name-value-unit tuple sequence from the name-value-unit tuple.*

Parameter:

None.

Java Calling Sequence:

```
NameValueSeqUnit nameValueSeqUnit = nameValueUnitSequenceIterator.nextOne();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.23.5 reset of interface NameValueUnitSequenceIterator

Purpose:

Reset the pointer in the element iterator to the first element.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
nameValueUnitSequenceIterator.reset();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.24 Query

Package:

org.asam.ods

Description:

The ASAM ODS interface Query.

Note:

This interface is not implemented, because the definition of the ASAM ODS query language is not completed yet.

4.24.1 executeQuery of interface Query

Purpose:

Execute query in asynchronous mode.

Note:

Not yet implemented, Query with query language not well defined in ASAM ODS.

Return:

void

Parameter:

[NameValueSequence](#) params (in) *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 aren'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);
```

Errors:

```
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
```

4.24.2 getInstance of interface Query**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.

Note:

Not yet implemented, Query with query language not well defined in ASAM ODS.

Return:

[InstanceElementIterator](#) result *The result of the query as an instance element iterator.*

Parameter:

None.

Java Calling Sequence:

```
InstanceElementIterator result = query.getInstance();
```

Errors:

```
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
```

4.24.3 getQueryEvaluator of interface Query**Purpose:**

Get the QueryEvaluator object which is responsible for this query.

Note:

Not yet implemented, Query with query language not well defined in ASAM ODS.

Return:

[QueryEvaluator](#) queryEvl *The QueryEvaluator object which is responsible for this query.*

Parameter:

None.

Java Calling Sequence:

```
QueryEvaluator queryEvl = query.getQueryEvaluator();
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.24.4 getStatus of interface Query**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.

Note:

Not yet implemented, Query with query language not well defined in ASAM ODS.

Return:

[QueryStatus](#) status *The status of the query.*

Parameter:

None.

Java Calling Sequence:

```
QueryStatus status = query.getStatus();
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.24.5 getTable of interface Query**Purpose:**

Get the query result. This method should only be called after the query has been executed. It returns the result as an structure.

Note:

Not yet implemented, Query with query language not well defined in ASAM ODS.

Return:

[NameValueSeqUnitSequence](#) result *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.*

Parameter:

None.

Java Calling Sequence:

```
NameValueSeqUnit[] result = query.getTable();
```

Errors:

[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](#)

4.24.6 getTableRows of interface Query**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.

Note:

Not yet implemented, Query with query language not well defined in ASAM ODS.

Return:

[NameValueUnitSequenceIterator](#) result *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.*

Parameter:

None.

Java Calling Sequence:

```
NameValueUnitSequenceIterator result = query.getTableRows();
```

Errors:

[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](#)

4.24.7 prepareQuery of interface Query

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.

Note:

Not yet implemented, Query with query language not well defined in ASAM ODS.

Return:

void

Parameter:

[NameValueSequence](#) params (in) *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

Java Calling Sequence:

```
query.prepareQuery(params);
```

Errors:

[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](#)

4.25 QueryEvaluator

Package:

org.asam.ods

Description:

The ASAM ODS query evaluator interface allows to perform queries in synchronous mode and to create Query objects for asynchronous execution.

Note:

This interface is not implemented, because the definition of the ASAM ODS query language is not completed yet.



4.25.1 createQuery of interface QueryEvaluator

Purpose:

Create a query object to execute it in asynchronous mode.

Note:

Not yet implemented, Query with query language not well defined in ASAM ODS.

Return:

[Query](#) queryObj *The query object.*

Parameter:

[T_STRING](#) queryStr (in) *The query string*

[NameValueSequence](#) params (in) *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);
```

Errors:

[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](#)

4.25.2 getInstances of interface QueryEvaluator

Purpose:

Evaluate a query in synchronous mode.

Note:

Not yet implemented, Query with query language not well defined in ASAM ODS.

Return:

[InstanceElementIterator](#) result *The result of the query as an instance element iterator.*

Parameter:

[T_STRING](#) queryStr (in) *The query string.*

NameValueSequence params (in) *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:

```
InstanceElementIterator result = queryEvaluator.getInstance(queryStr,params);
```

Errors:

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

4.25.3 getTable of interface QueryEvaluator

Purpose:

Evaluate a query in synchronous mode.

Note:

Not yet implemented, Query with query language not well defined in ASAM ODS.

Return:

NameValueSeqUnitSequence result *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.*

Parameter:

T_STRING queryStr (in) *The query string.*

NameValueSequence params (in) *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);
```

Errors:

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

4.25.4 getTableRows of interface QueryEvaluator

Purpose:

Evaluate a query in synchronous mode.

Note:

Not yet implemented, Query with query language not well defined in ASAM ODS.

Return:

[NameValueUnitSequenceIterator](#) result *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.*

Parameter:

[T_STRING](#) queryStr (in) *The query string.*

[NameValueSequence](#) params (in) *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:

```
NameValueUnitSequenceIterator result = queryEvaluator.getTableRows(queryStr,params);
```

Errors:

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

4.26 SMatLink

Package:

org.asam.ods

Description:

The ASAM ODS submatrix link interface.

Note:

This interface is not implemented, because the definition submatrix links are not completed yet.

4.26.1 getLinkType of interface SMatLink**Purpose:**

Get the link or build type.

Note:

Not yet implemented, SMatLink are not well defined in ASAM ODS.

Return:

[BuildUpFunction](#) linkType *The link type.*

Parameter:

None.

Java Calling Sequence:

```
BuildUpFunction linkType = sMatLink.getLinkType();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.26.2 getOrdinalNumber of interface SMatLink**Purpose:**

Get the ordinal or sequence number

Note:

Not yet implemented, SMatLink are not well defined in ASAM ODS.

Return:

[T_LONG](#) ordinalNumber *The sequence number.*

Parameter:

None.

Java Calling Sequence:

```
T_LONG ordinalNumber = sMatLink.getOrdinalNumber();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.26.3 getSMat1 of interface SMatLink

Purpose:

Get the first submatrix of the link.

Note:

Not yet implemented, SMatLink are not well defined in ASAM ODS.

Return:

[SubMatrix](#) subMatrix *The first submatrix of the link.*

Parameter:

None.

Java Calling Sequence:

```
SubMatrix subMatrix = sMatLink.getSMat1();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.26.4 getSMat1Columns of interface SMatLink

Purpose:

Get the bind columns of the first submatrix used in the link (e.g. Time).

Note:

Not yet implemented, SMatLink are not well defined in ASAM ODS.

Return:

[ColumnSequence](#) columns *The columns of the first submatrix.*

Parameter:

None.

Java Calling Sequence:

```
Column[] columns = sMatLink.getSMat1Columns();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.26.5 getSMat2 of interface SMatLink

Purpose:

Get the second submatrix of the link.

Note:

Not yet implemented, SMatLink are not well defined in ASAM ODS.

Return:

[SubMatrix](#) subMatrix *The second submatrix of the link.*

Parameter:

None.

Java Calling Sequence:

```
SubMatrix subMatrix = sMatLink.getSMat2();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.26.6 getSMat2Columns of interface SMatLink

Purpose:

Get the bind columns of the second submatrix used in the link (e.g. Time).

Note:

Not yet implemented, SMatLink are not well defined in ASAM ODS.

Return:

[ColumnSequence](#) columns *The columns of the second submatrix.*

Parameter:

None.

Java Calling Sequence:

```
Column[] columns = sMatLink.getSMat2Columns();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.26.7 setLinkType of interface SMatLink

Purpose:

Set the build or link type.

Note:

Not yet implemented, SMatLink are not well defined in ASAM ODS.

Return:

void

Parameter:

[BuildUpFunction](#) linkType (in) *The requested build-up function.*

Java Calling Sequence:

```
sMatLink.setLinkType(linkType);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_BUILDUP_FUNCTION](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.26.8 setOrdinalNumber of interface SMatLink

Purpose:

Set the ordinal or sequence number.

Note:

Not yet implemented, SMatLink are not well defined in ASAM ODS.

Return:

void

Parameter:

[T_LONG](#) ordinalNumber (in) *The sequence number.*

Java Calling Sequence:

```
sMatLink.setOrdinalNumber(ordinalNumber);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_ORDINALNUMBER](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.26.9 setSMat1 of interface SMatLink

Purpose:

Set the first submatrix of the link.

Note:

Not yet implemented, SMatLink are not well defined in ASAM ODS.

Return:

void

Parameter:

[SubMatrix](#) subMat1 (in) *The first submatrix of the submatrix link.*

Java Calling Sequence:

```
sMatLink.setSMat1(subMatrix);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_SUBMATRIX](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.26.10 setSMat1Columns of interface SMatLink

Purpose:

Set the bind columns of the first submatrix used in the link (e.g. Time).

Note:

Not yet implemented, SMatLink are not well defined in ASAM ODS.

Return:

void

Parameter:

[ColumnSequence](#) columns (in) *The column sequence of the submatrix.*

Java Calling Sequence:

```
sMatLink.setSMat1Columns(columns);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_COLUMN](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.26.11 setSMat2 of interface SMatLink

Purpose:

Set the second submatrix of the link.

Note:

Not yet implemented, SMatLink are not well defined in ASAM ODS.

Return:

void

Parameter:

[SubMatrix](#) subMat2 (in) *The second submatrix of the submatrix link.*

Java Calling Sequence:

```
sMatLink.setSMat2(subMatrix);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_INVALID_SUBMATRIX](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.26.12 setSMat2Columns of interface SMatLink

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.

Note:

Not yet implemented, SMatLink are not well defined in ASAM ODS.

Return:

void

Parameter:

[ColumnSequence](#) columns (in) *The column sequence of the submatrix.*

Java Calling Sequence:

```
sMatLink.setSMat2Columns(columns);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_INVALID_COLUMN](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.27 SubMatrix

Package:

org.asam.ods

Description:

The ASAM ODS submatrix interface. The instances of the submatrix can be accessed via the method `getRelatedInstances` of the interface `Measurement`.

Derived:

[InstanceElement](#)

4.27.1 `getColumns` of interface `SubMatrix`

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.

Return:

[ColumnSequence](#) columns *The columns of the submatrix.*

Parameter:

[Pattern](#) colPattern (in) *The name or the search pattern for the column names.*

Java Calling Sequence:

```
Column[] columns = subMatrix.getColumns(colPattern);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.MultiSessionTest.threadCreateValueMatrixFromSm.run()  
test.highqsoft.avalon.ValueMatrixTest.testCompareColumns()  
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromMeaSm()  
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromSm()
```

4.27.2 `getValueMatrix` of interface `SubMatrix`

Purpose:

Get a value matrix of the submatrix.



Return:

[ValueMatrix](#) valueMatrix *The value matrix.*

Parameter:

None.

Java Calling Sequence:

```
ValueMatrix valueMatrix = subMatrix.getValueMatrix();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_SMATLINK](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.MultiSessionTest.threadCompareValues.run()
test.highqsoft.avalon.MultiSessionTest.threadCreateValueMatrixFromSm.run()
test.highqsoft.avalon.ValueMatrixTest.testCompareColumns()
test.highqsoft.avalon.ValueMatrixTest.testCompareValues()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromMeaSm()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromSm()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromSmSelectMea()
test.highqsoft.avalon.ValueMatrixTest.testListColumns()
test.highqsoft.avalon.ValueMatrixTest.testValueMatrixGetMeaPoint()
test.highqsoft.avalon.ValueMatrixTest.testValueMatrixGetValue()
....
```

4.27.3 listColumns of interface SubMatrix**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.

Return:

[NameSequence](#) columnNames *The column names of the submatrix.*

Parameter:

[Pattern](#) colPattern (in) *The name or the search pattern for the column names.*

Java Calling Sequence:

```
Name[] columnNames = subMatrix.listColumns(colPattern);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.MultiSessionTest.threadCreateValueMatrixFromSm.run()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromSm()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromSmSelectMea()
test.highqsoft.avalon.ValueMatrixTest.testSubMatrixListColumns()
```

4.28 ValueMatrix

Package:

```
org.asam.ods
```

Description:

The ASAM ODS value matrix interface. Value matrix is an interface used by Measurement and SubMatrix to handle vectors of values.

It is only possible to set the values at a ValueMatrix, if the ValueMatrix is created from a submatrix instance element. If the valuematrix is created from a measurement instance element it is not allowed to set the values.

The following example shows how a valuematrix is create from a measurement with two submatrixes.

SubMatrix 1

Time	QuantityA
0.1	xxx1
0.2	xxx2
0.3	xxx3
0.4	xxx4

SubMatrix 2

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

Both submatrixes have the same independent column **Time**.

ValueMatrix

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
0.4	xxx4	

The server checks the independent column, if there is more than one submatrix at the measurement there must be exactly one independent column with the same measurement quantity in each

submatrix, otherwise the exception `AO_INVALID_VALUEMATRIX_STRUCTURE` is thrown.

The server throws the exception `AO_INVALID_VALUEMATRIX_STRUCTURE` when it is unable to create the valuematrix due to the data of the measurement

- if there are no independent column
- if there are different independent columns
- if there are submatrices with more than one independent column.

The server orders the values according to the values of the independent localcolumn. (in ascending order) It is up to the server what value is returned when a measurement quantity has two different values at the same independent point. The gaps are marked with Flags in `TS_ValueSeq` (flagvalue = 0).

The access to valuematrix 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' (rows) 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' (rows) 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' (rows) may be deleted from a submatrix.

There are three methods to set or get the values of a ValueMatrix.

Note:

The name of the column is the name of the instance of the measurement quantity. Due to the fact ASAM ODS requires the name of the local column must be identical with the name of the measurement quantity there will be normally no difference.

1. setValue or getValue, the following picture illustrate the area of the valuematrix which can be filled.

t	V	T	N	A	B	C
		X	X	X	X	
		X	X	X	X	

Figure 4.3: setValue

2. setValueVector or getValueVector, the following picture illustrate the area of the valuematrix which can be filled.

t	V	T	N	A	B	C
		X				
		X				
		X				
		X				
		X				

Figure 4.4: setValueVector

3. setMeaPoint or getMeaPoint, the following picture illustrate the area of the valuematrix which can be filled.

t	V	T	N	A	B	C
X	X	X	X	X	X	X

Figure 4.5: setMeaPoint

Note:

Names of the instances of AoMeasurementQuantity. The methods listColumns and getColumns of the interface ValuesMatrix have a parameter Pattern. With names of the instances of AoMeasurementQuantity (Eg "sin(x\500).(2-sin(x\10000))") the client have to escape the names otherwise the server will not be able to find the correct column.

An example how to use the ValueMatrix interface is given below.

```

/* @(#) $Id: UsingValueMatrix.java,v 1.8 2008/01/10 09:22:37 karst Exp $
*****
* COPYRIGHT I, 1996-2008
* Hans-Joachim Bothe, Andreas Hofmann, Karst Schaap, Michael Ziller
*****
* HighQSoft GmbH
* Schlossborner Weg 6b, D-61479 Glashuetten/Taunus, Germany
* Phone: +49 6174 62915, Fax: +49 6174 62935, Internet: www.highqsoft.com
*****
*
* All Rights Reserved.
*
* This software is the confidential and proprietary information of the
* authors. It may be freely copied and distributed with the following
* stipulations:
*
*   o No fee except to recover costs of media and delivery may
*     be charged for the use or possession of this software.
*   o Sources to this utility must be made available in machine-
*     readable form along with the executable form.
*   o No portion of this program may be used in any program sold
*     for a fee or used for production purposes.
*   o This copyright notice must not be removed.
*
* All brand names and product names used in this software are trademarks,
* registered trademarks, or trade names of their respective holders.
* The authors of this software are not associated with any product or
* vendor mentioned in the code or documentation.
*
*****
*/
/**
* ASAM ODS Example: Using ValueMatrix
*
* @author      Karst Schaap
* @version     $Revision: 1.8 $

```

```

* @since      $Date: 2008/01/10 09:22:37 $
*
* This example shows how to use the valuematrix interface
* <p>
* <TABLE BORDER class="PropertyTable">
* <TR><TD> InstanceId                                </TD>
*   <TD> Integer                                    </TD>
*   <TD> 1                                           </TD>
*   <TD> The Id of the instances of the
*     measurement application elements.<br>
*     <em>This value is an application property.</em> </TD></TR>
* <TR><TD> ColumnNamePattern                          </TD>
*   <TD> String                                      </TD>
*   <TD> "*"                                         </TD>
*   <TD> The pattern of the column names.<br>
*     <em>This value is an application property.</em> </TD></TR>
* </TABLE>
*/

import org.asam.ods.AoException;
import org.asam.ods.AoSession;
import org.asam.ods.ApplicationElement;
import org.asam.ods.ApplicationStructure;
import org.asam.ods.BaseElement;
import org.asam.ods.Column;
import org.asam.ods.DataType;
import org.asam.ods.InstanceElement;
import org.asam.ods.InstanceElementIterator;
import org.asam.ods.Measurement;
import org.asam.ods.NameValueSeqUnit;
import org.asam.ods.NameValueUnit;
import org.asam.ods.NameValueUnitIterator;
import org.asam.ods.Relationship;
import org.asam.ods.SetType;
import org.asam.ods.SubMatrix;
import org.asam.ods.TS_ValueSeq;
import org.asam.ods.T_LONGLONG;
import org.asam.ods.ValueMatrix;

import com.highqsoft.odsx.OdsxHelper;

public class UsingValueMatrix {

    static String columnPattern = "*";

    private static void usingValueMatrixPoints(ValueMatrix vm) throws AoException
    {
        System.out.println("usingValueMatrixPoints");
        int noValues = vm.getRowCount();
        int block=65536;
        Column[] cols = vm.getColumns(columnPattern);
        TS_ValueSeq val;
        int maxCols = 100;
        if (columnPattern.compareTo("*") == 0) {
            maxCols = 5;
        }

        /* Show all values of the first 5 columns */
        for (int valIndex = 0; valIndex < noValues; valIndex+=block)
        {
            for (int colIndex = 0; (colIndex < cols.length) && (colIndex < maxCols); colIndex++) {
                val = vm.getValueVector(cols[colIndex], valIndex, block);
                System.out.println("Values of column " + cols[colIndex].getName());
                System.out.println(OdsxHelper.ts_valueSeqToString(val));
            }
        }
    }
}

```



```

try {
    // Get the iterator to the values of the first row.
    NameValueUnitIterator nvuIter = vm.getValueMeaPoint(0);
    if (nvuIter != null) {
        int nValues = nvuIter.getCount();
        System.out.println("Number of values in row = " + nValues);

        // Any values in the row ?
        if (nValues > 0) {

            // Load a row in one step.
            NameValueUnit[] row = nvuIter.nextN(nValues);

            // Print all values in the row.
            for (int i=0; i<nValues && i < 100; i++) {
                System.out.println("    " + OdsxHelper.nameValueUnitToString(row[i]));
            }

        } else {
            System.err.println("No measurement values found!");
        }

        // Destroy name-value-unit iterator.
        nvuIter.destroy();

    } else {
        System.err.println("No value iterator available!");
    }

} catch (AoException aoException) {
    System.err.println("\nUsingValueMatrix, " + aoException.toString() + ":" +
        "minorCode " + aoException.minorCode +
        "\n    " + aoException.reason);
}

// Read all values at once of the first column.
val = vm.getValueVector(cols[0], 0, 0);
System.out.println("Values of column " + cols[0].getName());
System.out.println(OdsxHelper.ts_valueSeqToString(val));
}

private static void showValueMatrixPoints(ValueMatrix vm, int point, int count) throws AoException
{
    try {
        System.out.println("showValueMatrixPoints");
        // Get the iterator to the values of the .
        Column[] cols = vm.getColumns(columnPattern);
        NameValueSeqUnit[] nvu = vm.getValue(cols, point, count);
        // Print all values in the row.
        for (int i=0; i<nvu.length && i < 100; i++) {
            System.out.println("    " + OdsxHelper.nameValueSeqUnitToString(nvu[i]));
        }

    } catch (AoException aoException) {
        System.err.println("\nUsingValueMatrix, " + aoException.toString() + ":" +
            "minorCode " + aoException.minorCode +
            "\n    " + aoException.reason);
    }
}

public static Column[] removeIndepCols(Column[] cols, String[] indepCols) throws AoException {
    int resultLength = cols.length-indepCols.length;    // Length of result array
    Column[] notIndep = new Column[resultLength];        // Result array
    int wIndex=0;                                         // WriteIndex in result array
    boolean found;
    for (int colIndex = 0; colIndex < cols.length; colIndex++) {
        String colName = cols[colIndex].getName();
        found = false;

```

```

        for (int i = 0; i < indepCols.length; i++) {
            if (colName.compareTo(indepCols[i]) == 0) {
                found = true;
                System.out.println("Found independent column: " + colName);
            }
        }
        if ((!found) && (wIndex < resultLength)) {
            notIndep[wIndex++] = cols[colIndex];
        }
    }
    return(notIndep);
}

public static void modifyValueMatrixPoints(AoSession aos, ValueMatrix vm, int point) {
    try {
        System.out.println("modifyValueMatrixPoints");
        // Get the iterator to the values of the first row.
        Column[] cols = vm.getColumns(columnPattern);
        String[] indepCols = vm.listIndependentColumns("*");
        Column[] modCols = removeIndepCols(cols, indepCols);
        NameValueSeqUnit[] nvu = vm.getValue(modCols, point, 10);
        // Print all values in the row.
        for (int i=0; i<nvu.length && i < 100; i++) {
            System.out.println("    " + OdsxHelper.nameValueSeqUnitToString(nvu[i]));
        }
        // Modify the value
        for (int i=0; i<nvu.length; i++) {
            DataType dt = nvu[i].value.u.discriminator();
            switch (dt.value()) {
                case DataType._DT_DOUBLE:
                {
                    double[] f;
                    f = nvu[i].value.u.doubleVal();
                    for (int k = 0; k < f.length; k++) {
                        f[k] = f[k] + 0.5;
                    }
                    break;
                }
                case DataType._DT_FLOAT:
                {
                    float[] f;
                    f = nvu[i].value.u.floatVal();
                    for (int k = 0; k < f.length; k++) {
                        f[k] = f[k] + (float)0.5;
                    }
                    break;
                }
                case DataType._DT_LONG:
                {
                    int[] f;
                    f = nvu[i].value.u.longVal();
                    for (int k = 0; k < f.length; k++) {
                        f[k] = f[k] + 5;
                    }
                    break;
                }
                case DataType._DT_STRING:
                {
                    String[] f;
                    f = nvu[i].value.u.stringVal();
                    for (int k = 0; k < f.length; k++) {
                        f[k] = "String_" + k;
                    }
                    break;
                }
                case DataType._DT_DATE:
                {
                    String[] f;

```

```

        f = nvu[i].value.u.dateVal();
        for (int k = 0; k < f.length; k++) {
            f[k] = OdsxHelper.getOdsDate(null);
        }
        break;
    }
    default:
        System.out.println("Not implemented datatype: " + dt.value());
        break;
    }
}
System.out.println("modifyValueMatrixPoints new values");
for (int i=0; i<nvu.length && i < 100; i++) {
    System.out.println("    " + OdsxHelper.nameValueSeqUnitToString(nvu[i]));
}
aos.startTransaction();
try {
    vm.setValue(SetType.UPDATE, point, nvu);
    aos.commitTransaction();
} catch (AoException aoException) {
    aos.abortTransaction();
    System.err.println("\nUsingValueMatrix, " + aoException.toString() + ":" +
        "minorCode " + aoException.minorCode +
        "\n    " + aoException.reason);
}
} catch (AoException aoException) {
    System.err.println("\nUsingValueMatrix, " + aoException.toString() + ":" +
        "minorCode " + aoException.minorCode +
        "\n    " + aoException.reason);
}
}

public static void insertValueMatrixPoints(AoSession aos, ValueMatrix vm, int point) {
    try {
        System.out.println("insertValueMatrixPoints");
        // Get the iterator to the values of the first row.
        Column[] cols = vm.getColumns(columnPattern);
        String[] indepCols = vm.listIndependentColumns("*");
        Column[] insCols = removeIndepCols(cols, indepCols);
        NameValueSeqUnit[] nvu = vm.getValue(insCols, point, 3);
        // Print all values in the row.
        for (int i=0; i<nvu.length && i < 100; i++) {
            System.out.println("    " + OdsxHelper.nameValueSeqUnitToString(nvu[i]));
        }
        // Modify the value
        for (int i=0; i<nvu.length; i++) {
            DataType dt = nvu[i].value.u.discriminator();
            switch (dt.value()) {
                case DataType._DT_DOUBLE:
                {
                    double[] f;
                    f = nvu[i].value.u.doubleVal();
                    for (int k = 0; k < f.length; k++) {
                        f[k] = f[k] + 0.5;
                    }
                    break;
                }
                case DataType._DT_FLOAT:
                {
                    float[] f;
                    f = nvu[i].value.u.floatVal();
                    for (int k = 0; k < f.length; k++) {
                        f[k] = f[k] + (float)0.5;
                    }
                    break;
                }
                case DataType._DT_LONG:
            }
        }
    }
}

```

```

        {
            int[] f;
            f = nvu[i].value.u.longVal();
            for (int k = 0; k < f.length; k++) {
                f[k] = f[k] + 5;
            }
            break;
        }
        case DataType._DT_STRING:
        {
            String[] f;
            f = nvu[i].value.u.stringVal();
            for (int k = 0; k < f.length; k++) {
                f[k] = "String_" + k;
            }
            break;
        }
        case DataType._DT_DATE:
        {
            String[] f;
            f = nvu[i].value.u.dateVal();
            for (int k = 0; k < f.length; k++) {
                f[k] = OdsxHelper.getOdsDate(null);
            }
            break;
        }
        default:
            System.out.println("Not implemented datatype: " + dt.value());
            break;
    }
}
System.out.println("insertValueMatrixPoints new values");
for (int i=0; i<nvu.length && i < 100; i++) {
    System.out.println("    " + OdsxHelper.nameValueSeqUnitToString(nvu[i]));
}
aos.startTransaction();
try {
    vm.setValue(SetType.INSERT, point, nvu);
    aos.commitTransaction();
} catch (AoException aoException) {
    aos.abortTransaction();
    System.err.println("\nUsingValueMatrix, " + aoException.toString() + ":" +
        "minorCode " + aoException.minorCode +
        "\n    " + aoException.reason);
}
} catch (AoException aoException) {
    System.err.println("\nUsingValueMatrix, " + aoException.toString() + ":" +
        "minorCode " + aoException.minorCode +
        "\n    " + aoException.reason);
}
}

// Java application entry.
public static void main(String[] args) {

    // Say in which example we are.
    System.out.println("\nExample UsingValueMatrix...");
    System.out.println("-----");

    // Timestamp for performance check.
    long mSeconds = System.currentTimeMillis();

    try {

        // Establish a session to the service (without session options).
        AoSession aoSession = ExamplesConnect.connectService(args);
    }
}

```

```

try {
    // Get the application structure.
    ApplicationStructure as = aoSession.getApplicationStructure();

    // Get all application elements of type Measurement.
    ApplicationElement[] aeMea = as.getElementsByBaseType("AoMeasurement");

    // At least one measurement object found ?
    if (aeMea != null && aeMea.length > 0) {

        String ieIdStr = ExamplesConnect.getArgumentValue("InstanceId", "1");
        columnPattern = ExamplesConnect.getArgumentValue("ColumnNamePattern", "*");
        Integer ieIdLow = new Integer(ieIdStr);
        T_LONGLONG ieId;
        ieId = new T_LONGLONG();
        ieId.high = 0;
        ieId.low = ieIdLow.intValue();

        // Get the instances of the first measurement type.
        InstanceElement ieObj = aeMea[0].getInstanceById(ieId);

        Measurement mea = ieObj.upcastMeasurement();

        if (mea != null) {

            // Show the measurement instance name.
            System.out.println("Name of " + aeMea[0].getName() + " instance = " + mea.getName());

            try {
                // Get the value matrix of the measurement.
                ValueMatrix vm = mea.getValueMatrix();

                // Measurement values available ?
                if (vm != null) {

                    usingValueMatrixPoints(vm);
                    showValueMatrixPoints(vm, 2, 5);

                } else {
                    System.err.println("No measurement value matrix found!");
                }
            } catch (AoException aoException) {
                System.err.println("\nUsingValueMatrix, " + aoException.toString() + ":" +
                    "minorCode " + aoException.minorCode +
                    "\n    " + aoException.reason);
            }

            InstanceElement ieSm = null;
            InstanceElementIterator ieSmIter = ieObj.getRelatedInstancesByRelationship(Relationship.CHILD, "*");
            do {
                ieSm = ieSmIter.nextOne();
                if (ieSm != null) {
                    // The instance can be an AoSubMatrix or an AoMeasurementQuantiy
                    ApplicationElement aeSm = ieSm.getApplicationElement();
                    BaseElement beSm = aeSm.getBaseElement();
                    if (beSm.getType().compareToIgnoreCase("AoSubMatrix") == 0) {
                        SubMatrix sm = ieSm.upcastSubMatrix();
                        ValueMatrix vm = sm.getValueMatrix();
                        showValueMatrixPoints(vm, 2, 5);
                        modifyValueMatrixPoints(aoSession, vm, 15);
                        showValueMatrixPoints(vm, 2, 10);
                        insertValueMatrixPoints(aoSession, vm, 3);
                        showValueMatrixPoints(vm, 2, 10);
                    }
                    ieSm.destroy();
                }
            } while (ieSm != null);
        }
    }
}

```

```

        ieSmIter.destroy();

    } else {
        System.err.println("No measurements found!");
    }

    } else {
        System.err.println("No measurement types found!");
    }

    } catch (AoException aoException) {
        System.err.println("\nUsingValueMatrix, " + aoException.toString() + ":" +
            " minorCode " + aoException.minorCode +
            "\n    " + aoException.reason);
    }
    // Close the active session.
    aoSession.close();

    } catch (AoException aoException) {
        System.err.println("\nUsingValueMatrix, " + aoException.toString() + ":" +
            "\n    " + aoException.reason);
    }

    // Print elapsed time.
    System.out.println ("\nElapsed Time (in ms): " + (System.currentTimeMillis()-mSeconds));

    // Exit the application.
    System.exit(0);

}
}

```

4.28.1 addColumn of interface ValueMatrix

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.

Note:

See also **Transaction handling in ODS API**(p. 14). The name of the column is the name of the instance of the measurement quantity. Due to the fact ASAM ODS requires the name of the local column must be identical with the name of the measurement quantity there will be normally no difference.

Return:

[Column](#) column *The new column.*

Parameter:

[NameUnit](#) newColumn (in) *The name and unit of the column to add.*

Java Calling Sequence:

```
valueMatrix.addColumn(newColumn);
```



Errors:

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_TRANSACTION_NOT_ACTIVE

4.28.2 addColumnScaledBy of interface ValueMatrix**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.

Note:

Not yet implemented. See also **Transaction handling in ODS API**.(p. 14)

Return:

Column column *The new column.*

Parameter:

NameUnit newColumn (in) *The name and unit of the column to add.*

Column scalingColumn (in) *The scaling column.*

Java Calling Sequence:

```
Column column = valueMatrix.addColumnScaledBy(newColumn,scalingColumn);
```

Errors:

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
 AO_TRANSACTION_NOT_ACTIVE

4.28.3 destroy of interface ValueMatrix

Purpose:

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.

Note:

This method have only influence when corba is used.

Return:

void

Parameter:

None.

Java Calling Sequence:

```
vmObj.destroy()
```

Since:

ASAM ODS 5.0

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_TRANSACTION_NOT_ACTIVE](#)

Tested by:

```

test.highqsoft.avalon.MultiSessionTest.threadCompareValues.run()
test.highqsoft.avalon.MultiSessionTest.threadCreateValueMatrix.run()
test.highqsoft.avalon.MultiSessionTest.threadCreateValueMatrixFromSm.run()
test.highqsoft.avalon.ValueMatrixTest.testCompareValues()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrix()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromMeaSm()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromSm()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromSmSelectMea()
test.highqsoft.avalon.ValueMatrixTest.testUnitConversion()
test.highqsoft.avalon.ValueMatrixTest.testValueMatrixGetMeaPoint()
....

```

4.28.4 getColumnCount of interface ValueMatrix

Purpose:

Get the column count of the value matrix.

Return:

[T_LONG](#) columnCount *The number of columns of the value matrix.*



Parameter:

None.

Java Calling Sequence:

```
T_LONG columnCount = valueMatrix.getColumnCount();
```

Errors:

AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

```
test.highqsoft.odsapi.CreateInstanceTestCase.checkMeasurementInstances()  
test.highqsoft.odsapi.CreateMeasurementTest.testAddColumn()
```

4.28.5 getColumns of interface ValueMatrix

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.

Note:

The name of the column is the name of the instance of the measurement quantity and the pattern must match this names.. Due to the fact ASAM ODS requires the name of the local column must be identical with the name of the measurement quantity there will be normally no difference.

Return:

[ColumnSequence](#) columns *The columns of the value matrix, no matter whether the column is dependent, independent or scaling*

Parameter:

[Pattern](#) colPattern (in) *The name or the search pattern for the column names.*

Java Calling Sequence:

```
Column[] columns = valueMatrix.getColumns(colPattern);
```

Errors:

AO_BAD_PARAMETER
AO_CONNECTION_LOST
AO_IMPLEMENTATION_PROBLEM
AO_NOT_IMPLEMENTED
AO_NO_MEMORY
AO_SESSION_NOT_ACTIVE

Tested by:

```
test.highqsoft.avalon.MultiSessionTest.threadCompareValues.run()  
test.highqsoft.avalon.MultiSessionTest.threadCreateValueMatrix.run()  
test.highqsoft.avalon.ValueMatrixTest.testCheckUnit()  
test.highqsoft.avalon.ValueMatrixTest.testCompareColumns()
```

```

test.highqsoft.avalon.ValueMatrixTest.testCompareValues()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrix()
test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromMeaSm()
test.highqsoft.avalon.ValueMatrixTest.testListColumns()
test.highqsoft.avalon.ValueMatrixTest.testUnitConversion()
test.highqsoft.avalon.ValueMatrixTest.testValueMatrixGetValue()
....

```

4.28.6 getColumnScaledBy of interface ValueMatrix

Purpose:

Get the columns which are scaled by the given column.

Return:

[ColumnSequence](#) columns *The columns which are scaled by the given input column.*

Parameter:

[Column](#) scalingColumn (in) *The scaling column.*

Java Calling Sequence:

```
Column[] columns = valueMatrix.getColumnScaledBy(scalingColumn);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_NO_SCALING_COLUMN](#)

4.28.7 getIndependentColumns of interface ValueMatrix

Purpose:

Get the independent columns of the valuematrix. The independent columns are the columns used to build the valuematrix.

Note:

The name of the column is the name of the instance of the measurement quantity and the pattern must match this names.. Due to the fact ASAM ODS requires the name of the local column must be identical with the name of the measurement quantity there will be normally no difference.

Return:

[ColumnSequence](#) columns *The independent column of the value matrix.*

Parameter:

[Pattern](#) colPattern (in) *The name or the search pattern for the independent column name.*

Java Calling Sequence:

```
Column[] columns = valueMatrix.getIndependentColumns(colPattern);
```



Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.MultiSessionTest.threadCreateValueMatrixFromSm.run()
 test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromSm()
 test.highqsoft.odsapi.read.MeasurementTest.testValueMatrixIndep()

4.28.8 getRowCount of interface ValueMatrix**Purpose:**

Get the row count of the value matrix.

Return:

[T_LONG](#) rowCount *The number of rows of the value matrix.*

Parameter:

None.

Java Calling Sequence:

```
T_LONG rowCount = valueMatrix.getRowCount();
```

Errors:

[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.MultiSessionTest.threadCompareValues.run()
 test.highqsoft.avalon.ValueMatrixTest.testCompareValues()
 test.highqsoft.avalon.ValueMatrixTest.testValueMatrixGetValue()
 test.highqsoft.avalon.ValueMatrixTest.testValueMatrixGetValueVector()
 test.highqsoft.avalon.ValueMatrixTest.testValueMatrixTime()
 test.highqsoft.odsapi.read.MeasurementTest.testValueMatrixPoint()
 test.highqsoft.odsapi.CreateInstanceTestCase.checkMeasurementInstances()
 test.highqsoft.odsapi.CreateInstanceTestCase.checkMeasurementInstances()
 test.highqsoft.odsapi.UpdateMeasurementTest.ThreadViewValues.run()
 test.highqsoft.odsapi.XATFCopyTest.testCompareValueMatrixValues()

4.28.9 getScalingColumns of interface ValueMatrix**Purpose:**

Get the scaling column of the valuematrix.

Return:

[ColumnSequence](#) columns *The scaling columns of the value matrix.*

Parameter:

[Pattern](#) colPattern (in) *The name or the search pattern for the scaling column name.*

Java Calling Sequence:

```
Column[] columns = valueMatrix.getScalingColumns(colPattern);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.28.10 getValue of interface ValueMatrix**Purpose:**

Get the values of different columns of the valuematrix.

Note:

The name of the column is the name of the instance of the measurement quantity. Due to the fact ASAM ODS requires the name of the local column must be identical with the name of the measurement quantity there will be normally no difference.

Return:

[NameValueSeqUnitSequence](#) values *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 requested sequence.*

Parameter:

[ColumnSequence](#) columns (in) *The requested columns.*

[T_LONG](#) startPoint (in) *The starting point in the column.*

[T_LONG](#) count (in) *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);
```

Since:

ASAM ODS 5.0

See also:

[getValueVector](#) of interface [ValueMatrix](#)
[getValueMeaPoint](#) of interface [ValueMatrix](#)



Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.ValueMatrixTest.testValueMatrixGetValue()

4.28.11 getValueMeaPoint of interface ValueMatrix**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.

Return:

[NameValueUnitIterator](#) valueMeaPoint *The requested measurement point.*

Parameter:

[T_LONG](#) meaPoint (in) *The measurement point.*

Java Calling Sequence:

```
NameValueUnitIterator valueMeaPoint = valueMatrix.getValueMeaPoint(measPoint);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.ValueMatrixTest.testValueMatrixGetMeaPoint()

4.28.12 getValueVector of interface ValueMatrix**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.

Return:

[TS_ValueSeq](#) valueVector *The requested column values of the value matrix.*

Parameter:

[Column](#) col (in) *The column to retrieve the values from.*

[T_LONG](#) startPoint (in) *The starting point in the column.*

[T_LONG](#) count (in) *The number of points to be retrieved.*

Java Calling Sequence:

```
TS_Value[] valueVector = valueMatrix.getValueVector(col,startPoint,count);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_COLUMN](#)
[AO_INVALID_COUNT](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highqsoft.avalon.MultiSessionTest.threadCompareValues.run()
test.highqsoft.avalon.ValueMatrixTest.testCompareValues()
test.highqsoft.avalon.ValueMatrixTest.testUnitConversion()
test.highqsoft.avalon.ValueMatrixTest.testValueMatrixGetValueVector()
test.highqsoft.avalon.ValueMatrixTest.testValueMatrixTime()
test.highqsoft.odsapi.read.MeasurementTest.testValueMatrixIndep()
test.highqsoft.odsapi.read.MeasurementTest.testValueMatrixPoint()
test.highqsoft.odsapi.read.MeasurementTest.testValueMatrixUnit()
test.highqsoft.odsapi.read.UnitTest.testValuematrixCheck()
test.highqsoft.odsapi.CreateInstanceTestCase.checkMeasurementInstances()
....
```

4.28.13 listColumns of interface ValueMatrix**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.

Return:

[NameSequence](#) columnNames *The column names of the value matrix, no matter whether the column is dependent, independent or scaled by another one.*

Parameter:

[Pattern](#) colPattern (in) *The name or the search pattern for the column names.*

Java Calling Sequence:

```
Name[] columnNames = valueMatrix.listColumns(colPattern);
```

Errors:

[AO_BAD_PARAMETER](#)



[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

test.highqsoft.avalon.ValueMatrixTest.testCreateValueMatrixFromSmSelectMea()
 test.highqsoft.avalon.ValueMatrixTest.testListColumns()
 test.highqsoft.odsapi.CreateMeasurementTest.testDeleteValues()

4.28.14 listColumnsScaledBy of interface ValueMatrix

Purpose:

List the names of the columns, which are scaled by the given column.

Note:

The name of the column is the name of the instance of the measurement quantity. Due to the fact ASAM ODS requires the name of the local column must be identical with the name of the measurement quantity there will be normally no difference.

Return:

[NameSequence](#) columnNames *The names of the columns which are scaled by the given input column.*

Parameter:

[Column](#) scalingColumn (in) *The scaling column.*

Java Calling Sequence:

```
Name[] columnNames = valueMatrix.listColumnsScaledBy(scalingColumn);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)
[AO_NO_SCALING_COLUMN](#)

4.28.15 listIndependentColumns of interface ValueMatrix

Purpose:

Get the names of the independent columns of the valuematrix. The independent columns are the columns used to build the valuematrix.

Return:

[NameSequence](#) columnNames *The names of the independent columns of the value matrix.*

Parameter:

[Pattern](#) colPattern (in) *The name or the search pattern for the independent column name.*

Java Calling Sequence:

```
Name[] columnNames = valueMatrix.listIndependentColumns(colPattern);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

Tested by:

```
test.highsoft.avalon.ValueMatrixTest.testCompareColumns()
```

4.28.16 listScalingColumns of interface ValueMatrix**Purpose:**

Get the names of the scaling columns of the valuematrix.

Return:

[NameSequence](#) columnNames *The names of the scaling columns of the value matrix.*

Parameter:

[Pattern](#) colPattern (in) *The name or the search pattern for the scaling column name.*

Java Calling Sequence:

```
Name[] columnNames = valueMatrix.listScalingColumns(colPattern);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.28.17 removeValueMeaPoint of interface ValueMatrix**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.

Note:

Only for valuematrix of a submatrix. Only when all column of the valuematrix are given. The transaction must be activated, but there is no transaction control. The modification of the values is directly stored in the data storage. The name of the column is the name of the instance of the measurement quantity and the names must match these names. Due to the fact ASAM ODS requires the name of the local column must be identical with the name of the measurement quantity there will be normally no difference.

Return:

void

Parameter:

[NameSequence](#) columnNames (in) *The columns from which the measurement points are to be removed.*

[T_LONG](#) meaPoint (in) *The measurement point to be removed.*

[T_LONG](#) count (in) *The number of points to be removed from each column.*

Java Calling Sequence:

```
valueMatrix.removeValueMeaPoint(columnNames,meaPoint,count);
```

Errors:

[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](#)

4.28.18 removeValueVector of interface ValueMatrix**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.

Note:

Not yet implemented. See also **Transaction handling in ODS API.**(p. 14)

Return:

void

Parameter:

[Column](#) col (in) *The column from which the values are to be removed.*

[T_LONG](#) startPoint (in) *The starting point for the value removal.*

[T_LONG](#) count (in) *The number of points to be removed from the column.*

Java Calling Sequence:

```
valueMatrix.removeValueVector(col,startPoint,count);
```

Errors:

[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

4.28.19 setValue of interface ValueMatrix

Purpose:

Create or modify a number of value vectors 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 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.

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.

The ValueMatrix interface handles the values but not the raw values, to modify the raw values the instances of localcolumn must be created, the base attribute sequence_representation, raw_datatype and the generation_parameter must be created before the values are modified.

The values must have the datatype identical with the raw_datatype, any other datatype will cause an AO_INVALID_DATATYPE exception.

Note:

The transaction must be activated. See also **Transaction handling in ODS API**.(p. 14). The name of the column is the name of the instance of the measurement quantity and the names of the parameter value must match these names. Due to the fact ASAM ODS requires the name of the local column must be identical with the name of the measurement quantity there will be normally no difference.

Return:

void

Parameter:

[SetType](#) set (in) *The set type.*

[T_LONG](#) startPoint (in) *The measurement point.*

[NameValueSeqUnitSequence](#) value (in) *The values to be inserted.*

Java Calling Sequence:

```
valueMatrix.setValue(set,startPoint,value);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_INVALID_COLUMN](#)
[AO_INVALID_SET_TYPE](#)
[AO_INVALID_DATATYPE](#)
[AO_IS_MEASUREMENT_MATRIX](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.28.20 setValueMeaPoint of interface ValueMatrix**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 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.

Note:

Not yet implemented. See also **Transaction handling in ODS API**.(p. 14) The name of the column is the name of the instance of the measurement quantity and the names of the parameter value must match these names. Due to the fact ASAM ODS requires the name of the local column must be identical with the name of the measurement quantity there will be normally no difference.

Return:

void

Parameter:

[SetType](#) set (in) *The set type.*

[T_LONG](#) meaPoint (in) *The measurement point.*

[NameValueSequence](#) value (in) *The values to be inserted.*

Java Calling Sequence:

```
valueMatrix.setValueMeaPoint(set,meaPoint,value);
```

Errors:

[AO_BAD_PARAMETER](#)
[AO_CONNECTION_LOST](#)
[AO_IMPLEMENTATION_PROBLEM](#)
[AO_IS_MEASUREMENT_MATRIX](#)
[AO_NOT_IMPLEMENTED](#)
[AO_NO_MEMORY](#)
[AO_SESSION_NOT_ACTIVE](#)

4.28.21 setValueVector of interface ValueMatrix

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.



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.

Note:

See also **Transaction handling in ODS API**.(p. 14)

Return:

void

Parameter:

Column col (in) *The column whose values are to be set.*

SetType set (in) *The set type.*

T_LONG startPoint (in) *The starting point for the new values.*

TS_ValueSeq value (in) *The values to be inserted.*

Java Calling Sequence:

```
valueMatrix.setValueVector(col,set,startPoint,value);
```

Errors:

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



Chapter 5

ASAM ODS Structures

5.1 Definitions of the ACL structure.

The ACL Structure

The access control list entry.

rights

is of datatype: [T_LONG](#)

The access rights of the requested object.

usergroupId

is of datatype: [T_LONGLONG](#)

The usergroup Id.

5.2 Definitions of the AIDName structure.

The AIDName Structure

AID - Name pair.

aaName

is of datatype: [Name](#)

The attribute, or measured quantity name.

aid

is of datatype: [T_LONGLONG](#)

The Id of the application element.

5.3 Definitions of the AIDNameUnitId structure.

The AIDNameUnitId Structure

AID - Name - UnitId tuple.

attr

is of datatype: [AIDName](#)

The attribute of the application element (aid, name).

unitId

is of datatype: [T_LONGLONG](#)

The unit of the attribute of the column. The unitId is the Id of instance element with the basetype AoUnit.

5.4 Definitions of the AIDNameValueSeqUnitId structure.

The AIDNameValueSeqUnitId Structure

AID - Name - Value - UnitId quartet. Multiple values for on attribute.

attr

is of datatype: [AIDName](#)

The attribute of the application element (aid, name).

unitId

is of datatype: [T_LONGLONG](#)

The unit of the attribute of the column. The unitId is the Id of instance element with the basetype AoUnit.

values

is of datatype: [TS_ValueSeq](#)

The column values with value flags.

5.5 Definitions of the AIDNameValueUnitId structure.

The AIDNameValueUnitId Structure

AID - Name - Value - UnitId quartet.

attr

is of datatype: [AIDName](#)

The attribute of the application element (aid, name).

unitId

is of datatype: [T_LONGLONG](#)

The unit of the attribute of the column. The unitId is the Id of instance element with the basetype AoUnit.

values

is of datatype: [TS_Value](#)

The attribute values with value flags.

5.6 Definitions of the ApplAttr structure.

The ApplAttr Structure

The application attribute information (metadata) definition. The same information is available at the interface ApplicationAttribute

aaName

is of datatype: [Name](#)

The application attribute name. The same name is returned by the method getName() of the ApplicationAttribute interface.

At the RPC API this information was stored in the field aaName of the structure AttrSeq and the request AOP_GetAttr.

baName

is of datatype: [Name](#)

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 the method getBaseAttribute() of the interface ApplicationAttribute.

At the RPC API this information was stored in the field baName of the structure AttrSeq and the request AOP_GetAttr.

dType

is of datatype: [DataType](#)

The attribute data type. The same data type is given by the method getDataType of the interface ApplicationAttribute.

At the RPC API this information was stored in the field aDataType of the structure AttrSeq and the request AOP_GetAttr.

isObligatory

is of datatype: [T_BOOLEAN](#)

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.

isUnique

is of datatype: [T_BOOLEAN](#)

The indicator for unique attributes. The same boolean is returned by the method isUnique() of the interface ApplicationAttribute.

length

is of datatype: [T_LONG](#)

The maximum possible length of values. The same length is returned by the method getLength() of the interface ApplicationAttribute.

unitId

is of datatype: [T_LONGLONG](#)

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 was stored in the field `aUnit` of the structure `AttrSeq` and the request `AOP_GetAttr`.

5.7 Definitions of the ApplElem structure.

The ApplElem Structure

The application element definition. The same information is available at the interface `ApplicationElement`.

aeName

is of datatype: [Name](#)

The application element name. The name is given also with the method `getName()` at the interface `ApplicationElement`.

At the RPC API this information was stored in the field `aiName` of the structure `ApplInfSeq` and the request `AOP_GetApplInf`.

aid

is of datatype: [T_LONGLONG](#)

The application element id. The id is given also with the method `getId()` at the interface `ApplicationElement`.

At the RPC API this information was stored in the field `aiAId` of the structure `ApplInfSeq` and the request `AOP_GetApplInf`.

attributes

is of datatype: [ApplAttrSequence](#)

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.

beName

is of datatype: [Name](#)

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 RPC API 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`.

5.8 Definitions of the ApplicationRelationInstanceElementSeq structure.

The ApplicationRelationInstanceElementSeq Structure

The application relation with the instances to create the relation with new instances.

applRel

is of datatype: [ApplicationRelation](#)

The application relation.

instances

is of datatype: [InstanceElementSequence](#)

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.

5.9 Definitions of the ApplicationStructureValue structure.

The ApplicationStructureValue Structure

Application structure values. All values of the entire application structure are stored in this structure and loaded to the Client on request.

At the RPC API this information delivered by the request AOP_GetApplInf and AOP_GetAttr for each application element.

applElems

is of datatype: [ApplElemSequence](#)

The list of application elements.

applRels

is of datatype: [ApplRelSequence](#)

The list of relations in the application structure. The list of ApplRel's contains distinct entries for relations and their inverses. The field invName contains the partner relation.

Both relation, the normal and the inverse relation are given in the sequence.

5.10 Definitions of the ApplRel structure.

The ApplRel Structure

The application relation info structure. The same information is available at the interface ApplicationRelation.

arName

is of datatype: [Name](#)

The relation name. The name is return with the method `getName()` of the interface `ApplicationRelation`.

At the RPC API this information was stored in the field `arName` of the structure `ApplRelSeq` and the request `AOP_GetApplInf`.

arRelationRange

is of datatype: [RelationRange](#)

The range of the relation. Range of the relation is not stored in the physical storage. The relation range is return at the method `getRelationRange()` of the interface `ApplicationRelation`.

arRelationType

is of datatype: [RelationType](#)

The type of the relation. Type of the relation is not stored in the physical storage. The relation type is return at the method `getRelationType()` of the interface `ApplicationRelation`.

brName

is of datatype: [Name](#)

Name of the base relation from the `elem1` to the `elem2`. The base relation is also not stored in the physical storage.

elem1

is of datatype: [T_LONGLONG](#)

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 was stored in the field `arAid1` of the structure `ApplRelSeq` and the request `AOP_GetApplInf`.

elem2

is of datatype: [T_LONGLONG](#)

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 was stored in the field `arAid2` of the structure `ApplRelSeq` and the request `AOP_GetApplInf`.

invBrName

is of datatype: [Name](#)

Name of the inverse base relation from the `elem2` to the `elem1`. The base relation is also not stored in the physical storage.

invName

is of datatype: [Name](#)

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.

invRelationRange

is of datatype: [RelationRange](#)

The inverse range of the relation. Range of the relation is not stored in the physical storage. The inverse relation range is return at the method `getInverseRelationRange()` of the interface `ApplicationRelation`.

5.11 Definitions of the AttrResultSet structure.

The AttrResultSet Structure

The results for one attribute. The result set for all attributes are given in the sequence of the result set.

attrValues

is of datatype: [NameValueSeqUnitId](#)

The first 'how_many' results. All values have the same AIDName and the same UnitId.

rest

is of datatype: [NameValueUnitIdIterator](#)

The rest of the results. The iterator provides access to the consecutive NameValueSeqUnitId-packages of the result set.

5.12 Definitions of the ElemId structure.

The ElemId Structure

Instance element Id. The unique description of an instance element.

aid

is of datatype: [T_LONGLONG](#)

The Id of the application element.

iid

is of datatype: [T_LONGLONG](#)

The Id of the instance element.

5.13 Definitions of the ElemResultSet structure.

The ElemResultSet Structure

The result set for one element. The result set for all elements are given in the sequence of the result set.

aid

is of datatype: [T_LONGLONG](#)

The Id of the application element.

attrValues

is of datatype: [AttrResultSetSequence](#)

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.

5.14 Definitions of the ElemResultSetExt structure.

The ElemResultSetExt Structure

Restult set of one application element.

aid

is of datatype: [T_LONGLONG](#)

The application element Id.

values

is of datatype: [NameValueSeqUnitIdSequence](#)

The attribute values of the instances of the given application element.

5.15 Definitions of the InitialRight structure.

The InitialRight Structure

An initial right.

refAid

is of datatype: [T_LONGLONG](#)

The referencing application element.

rights

is of datatype: [T_LONG](#)

The initial access rights of the requested object.

usergroupId

is of datatype: [T_LONGLONG](#)

The usergroup Id of the Initial right list.



5.16 Definitions of the JoinDef structure.

The JoinDef Structure

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.

fromAID

is of datatype: [T_LONGLONG](#)

joiningType

is of datatype: [JoinType](#)

refName

is of datatype: [Name](#)

toAID

is of datatype: [T_LONGLONG](#)

5.17 Definitions of the NameUnit structure.

The NameUnit Structure

The ASAM ODS name-unit tuple structure.

unit

is of datatype: [T_STRING](#)

Column unit as string.

valName

is of datatype: [Name](#)

Attribute name or measured quantity name.

5.18 Definitions of the NameValue structure.

The NameValue Structure

The ASAM ODS name-value pair structure.

valName

is of datatype: [Name](#)

Attribute name or measured quantity name.

value

is of datatype: [TS_Value](#)

Attribute value or column value (vector).

5.19 Definitions of the NameValueSeqUnit structure.

The NameValueSeqUnit Structure

The ASAM ODS name-value-unit tuple structure with a sequence of values.

unit

is of datatype: [T_STRING](#)

Column unit as string.

valName

is of datatype: [Name](#)

Column name or measured quantity name.

value

is of datatype: [TS_ValueSeq](#)

Column value (vector).

5.20 Definitions of the NameValueSeqUnitId structure.

The NameValueSeqUnitId Structure

The ASAM ODS name-value-unitId tuple structure with a sequence of values.

unitId

is of datatype: [T_LONGLONG](#)

Column unit as Id.

valName

is of datatype: [Name](#)

Column name or measured quantity name.

value

is of datatype: [TS_ValueSeq](#)

Column value (vector).

5.21 Definitions of the NameValueUnit structure.

The NameValueUnit Structure

The ASAM ODS name-value-unit tuple structure. This structure is identical with the NameValueUnitId, except the unit is given as a string instead of an Id.

unit

is of datatype: [T_STRING](#)

Attribute or column unit as string.

valName

is of datatype: [Name](#)

Attribute name or measured quantity name.

value

is of datatype: [TS_Value](#)

Attribute value or column value (vector).

5.22 Definitions of the NameValueUnitId structure.

The NameValueUnitId Structure

The ASAM ODS name-value-unitid tuple structure. This structure is identical with the NameValueUnit, except the unit is given as an Id instead of a string.

unitId

is of datatype: [T_LONGLONG](#)

Id of attribute or column unit.

valName

is of datatype: [Name](#)

Attribute name or measured quantity name.

value

is of datatype: [TS_Value](#)

Attribute value or column value (vector).

5.23 Definitions of the QueryStructure structure.

The QueryStructure Structure

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 independent 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.

anuSeq

is of datatype: [AIDNameUnitIdSequence](#)

The sequence of attributes to be reported. A pattern is accepted for the attribute name.

At the RPC API this information was stored in the fields applId and nuSeq of the structure GetValReq and the request AOP_GetVal. At the RPC API only one application element could be selected.

condSeq

is of datatype: [SelValueSequence](#)

The query condition sequence.

At the RPC API this information was stored in the field nsSeq of the structure GetValReq and the request AOP_GetVal.

operSeq

is of datatype: [SelOperatorSequence](#)

The query condition operator sequence.

At the RPC API was the operator always 'AND'.

orderBy

is of datatype: [SelOrderSequence](#)



The order by sequence. The order of the result set.
At the RPC API it was not possible to set the order.

relInst

is of datatype: [ElemId](#)

The related instance. (aid == 0 && iid == 0) means no related instance specified.
At the RPC API this information was stored in the field elemId of the structure GetValReq and the request AOP_GetVal.

relName

is of datatype: [Name](#)

Name of the relation.
At the RPC API this information was stored in the field refName of the structure GetValReq and the request AOP_GetVal.

5.24 Definitions of the QueryStructureExt structure.

The QueryStructureExt Structure

The extended query structure.

anuSeq

is of datatype: [SelAIDNameUnitIdSequence](#)

The sequence of attributes to be reported. A pattern is accepted for the attribute name.
At the RPC API this information was stored in the fields applId and nuSeq of the structure GetValReq and the request AOP_GetVal. At the RPC API only one application element could be selected.

condSeq

is of datatype: [SelItemSequence](#)

The query condition sequence.
At the RPC API this information was stored in the field nsSeq of the structure GetValReq and the request AOP_GetVal.

groupBy

is of datatype: [AIDNameSequence](#)

Defines the grouping attributes for a request, necessary if aggregate functions are defined in the SelAIDNameUnitIdSequence.

joinSeq

is of datatype: [JoinDefSequence](#)

Defined the join between the application elements.

orderBy

is of datatype: [SelOrderSequence](#)

The order by sequence. The order of the result set.
At the RPC API interface it was not possible to set the order.

5.25 Definitions of the RelationRange structure.

The RelationRange Structure

The ASAM ODS relation range structure.

max

is of datatype: [T_SHORT](#)

The maximum number in the range. -1 means MANY without a specified maximum number.

min

is of datatype: [T_SHORT](#)

The minimum number in the range.

5.26 Definitions of the ResultSetExt structure.

The ResultSetExt Structure

The Result set of the extended query. The iterator is for instance oriented access.

firstElems

is of datatype: [ElemResultSetExtSequence](#)

The sequence of the first how_many result elements.

restElems

is of datatype: [ElemResultSetExtSeqIterator](#)

the iterator, which allows to iterate above the result values, the attributes of one instance each iteration.

5.27 Definitions of the SelAIDNameUnitId structure.

The SelAIDNameUnitId Structure

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.

aggregate

is of datatype: [AggrFunc](#)

The aggregate function.

attr

is of datatype: [AIDName](#)

The attribute of the application element (aid, name).

unitId

is of datatype: [T_LONGLONG](#)

The unit of the attribute of the column. The unitId is the Id of instance element with the basetype AoUnit.

5.28 Definitions of the SelOrder structure.

The SelOrder Structure

Order criteria.

ascending

is of datatype: [T_BOOLEAN](#)

ascending order, FALSE means descending.

attr

is of datatype: [AIDName](#)

Attribute specification.

5.29 Definitions of the SelValue structure.

The SelValue Structure

Structure for name value query or attribute search conditions.

attr

is of datatype: [AIDNameValueUnitId](#)

The attribute specification with unit of the value.

oper

is of datatype: [SelOpcode](#)

The compare operator between the attribute and value.

value

is of datatype: [TS_Value](#)

Value for the condition.

5.30 Definitions of the SelValueExt structure.

The SelValueExt Structure

The attribute selection structure.

attr

is of datatype: [AIDNameUnitId](#)

The attribute specification with unit Id.

oper

is of datatype: [SelOpcode](#)

The compare operator between the attribute and value.

value

is of datatype: [TS_Value](#)

Value for the condition.

5.31 Definitions of the T_COMPLEX structure.

The T_COMPLEX Structure

The ASAM ODS complex data structure. This type is represented in the datatype enumeration by DT_COMPLEX.

i

is of datatype: [T_FLOAT](#)

The imaginary part of the complex number.

r

is of datatype: [T_FLOAT](#)

The real part of the complex number.

5.32 Definitions of the T_DCOMPLEX structure.

The T_DCOMPLEX Structure

The ASAM ODS double-precision complex data structure. This type is represented in the datatype enumeration by DT_DCOMPLEX.

i

is of datatype: [T_DOUBLE](#)

The imaginary part of the double precision complex number.

r

is of datatype: [T_DOUBLE](#)

The real part of the double precision complex number.

5.33 Definitions of the T_ExternalReference structure.

The T_ExternalReference Structure

The description of an 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.

description

is of datatype: [T_STRING](#)

Description of the external reference.

location

is of datatype: [T_STRING](#)

Location of the external reference. (asam path or URL)

contentType

is of datatype: [T_STRING](#)

Mime type of the external object.

5.34 Definitions of the T_LONGLONG structure.

The T_LONGLONG Structure

The ASAM ODS 64 bit integer structure. This type is represented in the datatype enumeration by DT_LONGLONG.

high

is of datatype: [T_LONG](#)

The most significant 32 bits of the 64 bit value.

low

is of datatype: [T_LONG](#)

The least significant 32 bits of the 64 bit value.

5.35 Definitions of the TS_Value structure.

The TS_Value Structure

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.

flag

is of datatype: [T_SHORT](#)

The value flags.

u

is of datatype: [TS_Union](#)

The value union for values of all known datatypes.

5.36 Definitions of the TS_ValueSeq structure.

The TS_ValueSeq Structure

A structure with sequences of a certain type. Using this union instead of sequence <TS_Value> gives much better performance.

flag

is of datatype: [S_SHORT](#)

See TS_Value flag.

u

is of datatype: [TS_UnionSeq](#)

The value union for values of all known datatypes.

Chapter 6

ASAM ODS Unions

6.1 Definitions of the SelItem union.

The SelItem Union

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.
Attribute: SelType

operator

Datatype: [SelOperator](#)
CaseName: [SEL_OPERATOR_TYPE](#)

value

Datatype: [SelValueExt](#)
CaseName: [SEL_VALUE_TYPE](#)

6.2 Definitions of the TS_Union union.

The TS_Union Union

The Union definition for all datatypes. Attribute: DataType

blobVal

Datatype: [T_BLOB](#)
CaseName: [DT_BLOB](#)

booleanSeq

Datatype: [S_BOOLEAN](#)
CaseName: [DS_BOOLEAN](#)

booleanVal

Datatype: [T_BOOLEAN](#)
CaseName: [DT_BOOLEAN](#)

byteSeq

Datatype: [S_BYTE](#)
CaseName: [DS_BYTE](#)

bytestrSeq

Datatype: [S_BYTESTR](#)
CaseName: [DS_BYTESTR](#)

bytestrVal

Datatype: [T_BYTESTR](#)
CaseName: [DT_BYTESTR](#)

byteVal

Datatype: [T_BYTE](#)
CaseName: [DT_BYTE](#)

complexSeq

Datatype: [S_COMPLEX](#)
CaseName: [DS_COMPLEX](#)

complexVal

Datatype: [T_COMPLEX](#)
CaseName: [DT_COMPLEX](#)

dateSeq

Datatype: [S_DATE](#)
CaseName: [DS_DATE](#)

dateVal

Datatype: [T_DATE](#)
CaseName: [DT_DATE](#)

dcomplexSeq

Datatype: [S_DCOMPLEX](#)
CaseName: [DS_DCOMPLEX](#)

dcomplexVal

Datatype: [T_DCOMPLEX](#)
CaseName: [DT_DCOMPLEX](#)

doubleSeq

Datatype: [S_DOUBLE](#)
CaseName: [DS_DOUBLE](#)

doubleVal

Datatype: [T_DOUBLE](#)
CaseName: [DT_DOUBLE](#)

enumSeq

Datatype: [S_LONG](#)
CaseName: [DS_ENUM](#)

enumVal

Datatype: [T_LONG](#)
CaseName: [DT_ENUM](#)

extRefSeqDatatype: [S_ExternalReference](#)CaseName: [DS_EXTERNALREFERENCE](#)**extRefVal**Datatype: [T_ExternalReference](#)CaseName: [DT_EXTERNALREFERENCE](#)**floatSeq**Datatype: [S_FLOAT](#)CaseName: [DS_FLOAT](#)**floatVal**Datatype: [T_FLOAT](#)CaseName: [DT_FLOAT](#)**longlongSeq**Datatype: [S_LONGLONG](#)CaseName: [DS_LONGLONG](#)**longlongVal**Datatype: [T_LONGLONG](#)CaseName: [DT_LONGLONG](#)**longSeq**Datatype: [S_LONG](#)CaseName: [DS_LONG](#)**longVal**Datatype: [T_LONG](#)CaseName: [DT_LONG](#)**shortSeq**Datatype: [S_SHORT](#)CaseName: [DS_SHORT](#)**shortVal**Datatype: [T_SHORT](#)CaseName: [DT_SHORT](#)**stringSeq**Datatype: [S_STRING](#)CaseName: [DS_STRING](#)**stringVal**Datatype: [T_STRING](#)CaseName: [DT_STRING](#)

6.3 Definitions of the TS_UnionSeq union.

The TS_UnionSeq Union

Define a union with sequences of a certain type. Using this union instead of sequence <TS_Union> gives much better performance. Attribute: DataType

blobValDatatype: [S_BLOB](#)CaseName: [DT_BLOB](#)**booleanSeq**Datatype: [SS_BOOLEAN](#)CaseName: [DS_BOOLEAN](#)**booleanVal**Datatype: [S_BOOLEAN](#)CaseName: [DT_BOOLEAN](#)**byteSeq**Datatype: [SS_BYTE](#)CaseName: [DS_BYTE](#)**bytestrSeq**Datatype: [SS_BYTESTR](#)CaseName: [DS_BYTESTR](#)**bytestrVal**Datatype: [S_BYTESTR](#)CaseName: [DT_BYTESTR](#)**byteVal**Datatype: [S_BYTE](#)CaseName: [DT_BYTE](#)**complexSeq**Datatype: [SS_COMPLEX](#)CaseName: [DS_COMPLEX](#)**complexVal**Datatype: [S_COMPLEX](#)CaseName: [DT_COMPLEX](#)**dateSeq**Datatype: [SS_DATE](#)CaseName: [DS_DATE](#)**dateVal**Datatype: [S_DATE](#)CaseName: [DT_DATE](#)**dcomplexSeq**Datatype: [SS_DCOMPLEX](#)CaseName: [DS_DCOMPLEX](#)**dcomplexVal**Datatype: [S_DCOMPLEX](#)CaseName: [DT_DCOMPLEX](#)

doubleSeq

Datatype: [SS_DOUBLE](#)
CaseName: [DS_DOUBLE](#)

doubleVal

Datatype: [S_DOUBLE](#)
CaseName: [DT_DOUBLE](#)

enumSeq

Datatype: [SS_LONG](#)
CaseName: [DS_ENUM](#)

enumVal

Datatype: [S_LONG](#)
CaseName: [DT_ENUM](#)

extRefSeq

Datatype: [SS_ExternalReference](#)
CaseName: [DS_EXTERNALREFERENCE](#)

extRefVal

Datatype: [S_ExternalReference](#)
CaseName: [DT_EXTERNALREFERENCE](#)

floatSeq

Datatype: [SS_FLOAT](#)
CaseName: [DS_FLOAT](#)

floatVal

Datatype: [S_FLOAT](#)
CaseName: [DT_FLOAT](#)

longlongSeq

Datatype: [SS_LONGLONG](#)
CaseName: [DS_LONGLONG](#)

longlongVal

Datatype: [S_LONGLONG](#)
CaseName: [DT_LONGLONG](#)

longSeq

Datatype: [SS_LONG](#)
CaseName: [DS_LONG](#)

longVal

Datatype: [S_LONG](#)
CaseName: [DT_LONG](#)

shortSeq

Datatype: [SS_SHORT](#)
CaseName: [DS_SHORT](#)

shortValDatatype: [S_SHORT](#)CaseName: [DT_SHORT](#)**stringSeq**Datatype: [SS_STRING](#)CaseName: [DS_STRING](#)**stringVal**Datatype: [S_STRING](#)CaseName: [DT_STRING](#)

Chapter 7

ASAM ODS Enumerations

7.1 Definitions of the AggrFunc enumeration.

The AggrFunc Enumeration

The supported aggregate functions of the GetInstanceExt.

NONE 0

No aggregate function is used for attribute.

COUNT 1

Count

DCOUNT 2

Distinct count

MIN 3

Min; only for numerical values

MAX 4

Max; only for numerical values

AVG 5

Average; only for numerical values

STDDEV 6

Standard deviation; only for numerical values

7.2 Definitions of the AttrType enumeration.

The AttrType Enumeration

The ASAM ODS attribute type codes.

APPLATTR_ONLY 0

Report only application attributes.

INSTATTR_ONLY 1

Report only instance attributes.

ALL 2

All attributes.

7.3 Definitions of the BuildUpFunction enumeration.

The BuildUpFunction Enumeration

The ASAM ODS build-up function codes for measurement views.

BUP_JOIN 0

Join the columns

BUP_MERGE 1

Merge the columns

BUP_SORT 2

Sort the columns

7.4 Definitions of the DataType enumeration.

The DataType Enumeration

The ASAM ODS data types.

DT_xxx Basic data types.

DS_xxx Sequence of basic data type.

||

|+- T == Type, S == Sequences.

+ - D == Datatype.

DT_UNKNOWN 0

Unknown datatype.

DT_STRING 1

String.

DT_SHORT 2

Short value (16 bit).

DT_FLOAT 3

Float value (32 bit).



DT_BOOLEAN 4

Boolean value.

DT_BYTE 5

Byte value (8 bit).

DT_LONG 6

Long value (32 bit).

DT_DOUBLE 7

Double precision float value (64 bit).

DT_LONGLONG 8

LongLong value (64 bit).

DT_ID 9

LongLong value (64 bit). Not used. DT_LONGLONG is used instead.

DT_DATE 10

Date. Meaning: YYYYMMDDhhmmsslllccnnn....

- YYYY = year, required.
- MM = month, optional.
- DD = day, optional.
- hh = hour, optional.
- mm = minute, optional.
- ss = second, optional.
- lll = millisec, optional, not supported by Oracle timestamp.
- ccc = microsec, optional, not supported by Oracle timestamp.
- nnn = nanosec, optional, not supported by Oracle timestamp.

DT_BYTESTR 11

Bytestream.

DT_BLOB 12

Blob.

DT_COMPLEX 13

Complex value (32 bit each part).

DT_DCOMPLEX 14

Complex value (64 bit each part).

DS_STRING 15

String sequence.

DS_SHORT 16

Short sequence.

DS_FLOAT 17

Float sequence.

DS_BOOLEAN 18

Boolean sequene.

DS_BYTE 19

Byte sequence.

DS_LONG 20

Long sequence.

DS_DOUBLE 21

Double sequence.

DS_LONGLONG 22

Longlong sequence.

DS_COMPLEX 23

Complex sequence.

DS_DCOMPLEX 24

Double complex sequence.

DS_ID 25

LongLong sequence. Not used. DS_LONGLONG is used instead.

DS_DATE 26

Date sequence.

DS_BYTESTR 27

Bytestream sequence.

DT_EXTERNALREFERENCE 28

External reference.

DS_EXTERNALREFERENCE 29

Sequence of external reference.

DT_ENUM 30

The enumeration datatype.

DS_ENUM 31

The enumeration sequence datatype.



7.5 Definitions of the ErrorCode enumeration.

The ErrorCode Enumeration

The ASAM ODS error codes.

AO_UNKNOWN_ERROR 0

Use the zero as unknown error to avoid confusing error messages if no error code has been set.

AO_ACCESS_DENIED 1

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 2

The BAD_OPERATION error code is used when a method is invalid in a marshalling operation.

AO_BAD_PARAMETER 3

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_CONNECT_FAILED 4

The connect to a server failed. This error may occur if the server is down or currently unreachable.

AO_CONNECT_REFUSED 5

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_CONNECTION_LOST 6

Due to a hardware or network software problem the connection to the server was lost.

AO_DUPLICATE_BASE_ATTRIBUTE 7

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 8

The implicit or explicit specified name is already in use but it is required to be unique.

AO_DUPLICATE_VALUE 9

The attribute is marked unique in the application model. Thus duplicate values are not allowed.

AO_HAS_INSTANCES 10

The operation is not allowed for elements that have instances.

AO_HAS_REFERENCES 11

The requested operation is not permitted because the target element has references.

AO_IMPLEMENTATION_PROBLEM 12

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 13

The units are incompatible. No conversion rule is known.

AO_INVALID_ASAM_PATH 14

The specified Asam path is invalid.

AO_INVALID_ATTRIBUTE_TYPE 15

The requested attribute type is invalid.

AO_INVALID_BASE_ELEMENT 16

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_BASETYPE 17

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_BUILDUP_FUNCTION 18

The specified build-up function is invalid.

AO_INVALID_COLUMN 19

The specified column is invalid.

AO_INVALID_COUNT 20

The specified number of points is invalid (probably negative).

AO_INVALID_DATATYPE 21

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 "Data-Type".

AO_INVALID_ELEMENT 22

The element is invalid in this context.

AO_INVALID_LENGTH 23

The given length is invalid. Negative length values are not allowed.

AO_INVALID_ORDINALNUMBER 24

The ordinal number is either already used or less than zero.

AO_INVALID_RELATION 25

The relation is invalid. The related elements and the base relation do not fit.

AO_INVALID_RELATION_RANGE 26

The specified relation range is invalid.

AO_INVALID_RELATION_TYPE 27

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 "Relation-Type".

AO_INVALID_RELATIONSHIP 28

This error may occur only in non-typesafe language bindings. To avoid this error in all language bindings it is recommended to use always the definitions of the enumeration "Relationship".

AO_INVALID_SET_TYPE 29

The specified set-type is invalid.

AO_INVALID_SMATLINK 30

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 31

The specified submatrix is invalid.

AO_IS_BASE_ATTRIBUTE 32

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 33

Properties of base relations may not be changed.

AO_IS_MEASUREMENT_MATRIX 34

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 35

A computation error occurred. This can be an overflow, an underflow or a division by zero.

AO_MISSING_APPLICATION_ELEMENT 36

A required application element is missing.

AO_MISSING_ATTRIBUTE 37

A required (obligatory) attribute is missing.

AO_MISSING_RELATION 38

A required relation is missing.

AO_MISSING_VALUE 39

An obligatory value is missing (the AO_VF_DEFINED flag is zero).

AO_NO_MEMORY 40

No more volatile memory available.

AO_NO_PATH_TO_ELEMENT 41

A free-floating element was detected. No navigation path leads to this element.

AO_NOT_FOUND 42

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 43

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 44

This error occurs if the instances of a property are required to be unique.

AO_OPEN_MODE_NOT_SUPPORTED 45

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_SESSION_LIMIT_REACHED 46

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 47

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 48

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 49

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_HAS_BASE_RELATION 50

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_BASE_ATTRIBUTE 51

Base attribute found. It is not allowed to modify the datatype, unique- or obligatory flag .

AO_UNKNOWN_UNIT 52

The unit is unknown.

AO_NO_SCALING_COLUMN 53

The column is no scaling column.

AO_QUERY_TYPE_INVALID 54

The server does not support the specified query language type.

AO_QUERY_INVALID 55

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_PROCESSING_ERROR 56

Some error occurred during the execution of the query.

AO_QUERY_TIMEOUT_EXCEEDED 57

It was not possible to execute the query within the time limit set by parameter "Max-Duration".

AO_QUERY_INCOMPLETE 58

The execution of the query was not yet completed.

AO_QUERY_INVALID_RESULTTYPE 59

The requested result type of the query do not match with the previous definition of the result type.

7.6 Definitions of the JoinType enumeration.

The JoinType Enumeration

The type of the join.

JTDEFAULT 0

Force inner join.

JTOUTER 1

Force outer join on destination AID.



7.7 Definitions of the QueryStatus enumeration.

The QueryStatus Enumeration

Status of the query execution.

COMPLETE 0

The execution is ready.

INCOMPLETE 1

The execution is still running.

7.8 Definitions of the Relationship enumeration.

The Relationship Enumeration

The ASAM ODS relationships.

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: supertype.

SUBTYPE 6

Inheritance relationship: subtype.

ALL_REL 7

Any of the relationships above.

7.9 Definitions of the RelationType enumeration.

The RelationType Enumeration

The ASAM ODS relation types.

FATHER_CHILD 0

Father-child realation.

INFO 1

Info relation.

INHERITANCE 2

Inheritance relation.

7.10 Definitions of the RightsSet enumeration.

The RightsSet Enumeration

The ASAM ODS types for setting access rights.

SET_RIGHT 0

Set the given rights, overwrite the existing rights.

ADD_RIGHT 1

Add the given rights to the existing rights.

REMOVE_RIGHT 2

Remove the given rights form the existing rights.

7.11 Definitions of the SelOpcode enumeration.

The SelOpcode Enumeration

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 `SCI_`.

EQ 0

Equal

NEQ 1

Not equal

LT 2

Less then

GT 3

Greater then

LTE 4

Less then equal

GTE 5

Greater then equal

INSET 6

In set, value can be a sequence.

NOTINSET 7

Not in set, value can be a sequence.

LIKE 8

like, use pattern matching, see Pattern for the wildcard definitions.

CI_EQ 9

Equal. case insensitive for DT_STRING.

CI_NEQ 10

Not equal. case insensitive for DT_STRING.

CI_LT 11

Less then. case insensitive for DT_STRING.

CI_GT 12

Greater then. case insensitive for DT_STRING.

CI_LTE 13

Less then equal. case insensitive for DT_STRING.

CI_GTE 14

Greater then equal. case insensitive for DT_STRING.

CI_INSET 15

In set, value can be a sequence. case insensitive for DT_STRING.

CI_NOTINSET 16

Not in set, value can be a sequence. case insensitive for DT_STRING.

CI_LIKE 17

like, use pattern matching, see Pattern for the wildcard definitions. case insensitive for DT_STRING.

IS_NULL 18

Value is NULL

IS_NOT_NULL 19

Value is not NULL

7.12 Definitions of the SelOperator enumeration.

The SelOperator Enumeration

Operator, bracket open and close.

AND 0

AND the two conditions.

OR 1

OR the two conditions.

NOT 2

Negate the next condition.

OPEN 3

Open brackets.

CLOSE 4

Close brackets

7.13 Definitions of the SelType enumeration.

The SelType Enumeration

The selection type.

SEL_VALUE_TYPE 0

Selection value.

SEL_OPERATOR_TYPE 1

Selection logical operator.

7.14 Definitions of the SetType enumeration.

The SetType Enumeration

The ASAM ODS types for setting values.

APPEND 0

Append data to the value matrix.

INSERT 1

Insert data into the value matrix.

UPDATE 2

Modify existing data of the value matrix.

REMOVE 3

Remove the given information.

7.15 Definitions of the SeverityFlag enumeration.

The SeverityFlag Enumeration

The ASAM ODS error severity flags.

SUCCESS 0

Ok.

INFORMATION 1

Information.

WARNING 2

Warning.

ERROR 3

Error.



Chapter 8

ASAM ODS Constants

8.1 Definitions of the LockMode constant.

The LockMode Constant

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.

LOCK_INSTANCEELEMENT

is of datatype [T_SHORT](#) with value 0

Lock the instance element. (Default LockMode)

LOCK_APPLICATIONELEMENT

is of datatype [T_SHORT](#) with value 1

Lock the application element, all instances of the application element are locked.

LOCK_CHILDREN

is of datatype [T_SHORT](#) with value 2

Lock the children of the locked object. This mode can be combined with one of the upper two modi.

8.2 Definitions of the QueryConstants constant.

The QueryConstants Constant

The ASAM ODS query constants.

MaxDurationDEFAULT

is of datatype [T_LONG](#) with value 0

Default value of max duration parameter of the query (no limitations).

MaxDuration

is of datatype [T_STRING](#) with value "MaxDuration"

The ASAM ODS max duration parameter of the query.

QueryResultType

is of datatype **T_STRING** with value "QueryResultType"

The ASAM ODS query result type parameter.

QueryResultTypeDEFAULT

is of datatype **T_LONG** with value ResultType::INSTELEM_ITERATOR_AS_RESULT

Default value of the ASAM ODS query result type parameter.

8.3 Definitions of the ResultType constant.

The ResultType Constant

The ASAM ODS query result types.

INSTELEM_ITERATOR_AS_RESULT

is of datatype **T_SHORT** with value 0

Iterator of instance elements as result of the query (the default).

TABLE_ITERATOR_AS_RESULT

is of datatype **T_SHORT** with value 1

Iterator for table access as result type of the query.

TABLE_AS_RESULT

is of datatype **T_SHORT** with value 2

Table as result type of the query.

8.4 Definitions of the SecurityLevel constant.

The SecurityLevel Constant

The security level of an application element.

NO_SECURITY

is of datatype **T_LONG** with value 0

No security defined.

ELEMENT_SECURITY

is of datatype **T_LONG** with value 1

Security scaled for the application element.



INSTANCE_SECURITY

is of datatype **T_LONG** with value 2

Security scaled for instance elements.

ATTRIBUTE_SECURITY

is of datatype **T_LONG** with value 4

Security scaled for application attributes.

8.5 Definitions of the SecurityRights constant.

The SecurityRights Constant

The bits of the security rights.

SEC_READ

is of datatype **T_LONG** with value 1

Read access is allowed.

SEC_UPDATE

is of datatype **T_LONG** with value 2

Update access to an existing object is allowed.

SEC_INSERT

is of datatype **T_LONG** with value 4

Creating new instances is allowed.

SEC_DELETE

is of datatype **T_LONG** with value 8

Delete of the object is allowed.

SEC_GRANT

is of datatype **T_LONG** with value 16

Access rights may be passed on.

Chapter 9

ASAM ODS Types

9.1 Definition of the CORBA types.

The CORBA types are part of the Corba specification and used by the ASAM ODS. There are no other Corba datatypes used by ASAM ODS.

string

The corba string.

boolean

The corba boolean.

octet

The corba 8 bit signed integer.

short

The corba 16 bit signed integer.

long

The corba 32 bit signed integer.

float

The corba 32 bit IEEE-Floating point number.

double

The corba 64 bit IEEE-Floating point number.

9.2 Definitions of the Scalar type.

The Scalar type

A 8 Byte floating point number. This type is represented in the datatype enumeration by DT_DOUBLE.

T_FLOAT

is of datatype [float](#)

A 4 Byte floating point number. This type is represented in the datatype enumeration by DT_FLOAT.

T_LONG

is of datatype [long](#)

A 4 Byte signed interger number. This type is represented in the datatype enumeration by DT_LONG.

T_SHORT

is of datatype [short](#)

A 2 Byte signed interger number. This type is represented in the datatype enumeration by DT_SHORT.

T_STRING

is of datatype [string](#)

A NUL-Terminated string. This type is represented in the datatype enumeration by DT_STRING.

9.3 Definitions of the Sequence type.

The Sequence type

ACLSequence

is of datatype [ACL](#)

The sequence of access control list entries for a requested object.

AIDNameSequence

is of datatype [AIDName](#)

Sequence of AID - Name pairs.

AIDNameUnitIdSequence

is of datatype [AIDNameUnitId](#)

Sequence of AID - Name - UnitId tuple.

AIDNameValueSeqUnitIdSequence

is of datatype [AIDNameValueSeqUnitId](#)

Sequence of AID - Name - UnitId tuple.

ApplAttrSequence

is of datatype [ApplAttr](#)

Sequence of ApplAttr.

ApplElemSequence

is of datatype [ApplElem](#)

Application element definition sequence. The application elements are given with the method `getElements()` at the interface `ApplicationStructure`.

ApplicationAttributeSequence

is of datatype [ApplicationAttribute](#)

Sequecne of ApplicationAttribute objects.

ApplicationElementSequence

is of datatype [ApplicationElement](#)

Sequence of ApplicationElement objects.

ApplicationRelationInstanceElementSeqSequence

is of datatype [ApplicationRelationInstanceElementSeq](#)

Sequence of the application relation with corresponding instances.

ApplicationRelationSequence

is of datatype [ApplicationRelation](#)

Sequence of ApplicationRelation objects.

ApplRelSequence

is of datatype [ApplRel](#)

Application relation sequence. The application relations are given with the method `getRelations()` at the interface `ApplicationStructure`.

AttrResultSetSequence

is of datatype [AttrResultSet](#)

Sequence of the attribute result sets.

BaseAttributeSequence

is of datatype [BaseAttribute](#)

Sequence of BaseAttribute objects.

BaseElementSequence

is of datatype [BaseElement](#)

Sequence of BaseElement objects.

BaseRelationSequence

is of datatype [BaseRelation](#)

Sequence of BaseRelation objects.

BaseTypeSequence

is of datatype [BaseType](#)

Sequence of BaseType.

ColumnSequence

is of datatype [Column](#)

Sequence of Column objects.

ElemIdSequence

is of datatype [ElemId](#)

Sequence of element Id's.

ElemResultSetExtSequence

is of datatype [ElemResultSetExt](#)

Sequence of ElemResultSetExt

ElemResultSetSequence

is of datatype [ElemResultSet](#)

Sequence of the element result sets.

InitialRightSequence

is of datatype [InitialRight](#)

The sequence of the initial rights.

InstanceElementSequence

is of datatype [InstanceElement](#)

Sequence of InstanceElement objects.

JoinDefSequence

is of datatype [JoinDef](#)

Sequence of the join definitions.

NameSequence

is of datatype [Name](#)

Sequence of Name.

NameValueSequence

is of datatype [NameValue](#)

Sequence of NameValue-pairs.

NameValueSeqUnitIdSequence

is of datatype [NameValueSeqUnitId](#)

Sequence of NameValueSeqUnitId-triple.

NameValueSeqUnitSequence

is of datatype [NameValueSeqUnit](#)

Sequence of NameValueSeqUnit-triple.

NameValueUnitSequence

is of datatype [NameValueUnit](#)

Sequence of NameValueUnit-triple.

ResultSetExtSequence

is of datatype [ResultSetExt](#)

The sequence of result set of the extended query, for each application element a result set.

S_BLOB

is of datatype [T_BLOB](#)

Sequence of blob objects. This type is represented in the datatype enumeration by DS_BLOB.

S_BOOLEAN

is of datatype [T_BOOLEAN](#)

Sequence of boolean values. This type is represented in the datatype enumeration by DS_BOOLEAN.

S_BYTE

is of datatype [T_BYTE](#)

Sequence of byte values. This type is represented in the datatype enumeration by DS_BYTE.

S_BYTESTR

is of datatype [T_BYTESTR](#)

Sequence of byte value sequences. This type is represented in the datatype enumeration by DS_BYTESTR.

S_COMPLEX

is of datatype [T_COMPLEX](#)

Sequence of float complex values. This type is represented in the datatype enumeration by DS_COMPLEX.

S_DATE

is of datatype [T_DATE](#)

Sequence of date values. This type is represented in the datatype enumeration by DS_DATE.

S_DCOMPLEX

is of datatype [T_DCOMPLEX](#)

Sequence of double complex values. This type is represented in the datatype enumeration by DS_DCOMPLEX.

S_ DOUBLE

is of datatype [T_ DOUBLE](#)

Sequence of double values. This type is represented in the datatype enumeration by DS_ - DOUBLE.

S_ ExternalReference

is of datatype [T_ ExternalReference](#)

Sequence of external references. This type is represented in the datatype enumeration by DS_ - EXTERNALREFERENCE.

S_ FLOAT

is of datatype [T_ FLOAT](#)

Sequence of float values. This type is represented in the datatype enumeration by DS_ - FLOAT.

S_ LONG

is of datatype [T_ LONG](#)

Sequence of long values. This type is represented in the datatype enumeration by DS_ - LONG.

S_ LONGLONG

is of datatype [T_ LONGLONG](#)

Sequence of LongLong values. This type is represented in the datatype enumeration by DS_ - LONGLONG.

S_ SHORT

is of datatype [T_ SHORT](#)

Sequence of short values. This type is represented in the datatype enumeration by DS_ - SHORT.

S_ STRING

is of datatype [T_ STRING](#)

Sequence of string values. This type is represented in the datatype enumeration by DS_ - STRING.

SelAIDNameUnitIdSequence

is of datatype [SelAIDNameUnitId](#)

Sequence of selected attributes.



SelItemSequence

is of datatype [SelItem](#)

The sequence of selection items.

SelOperatorSequence

is of datatype [SelOperator](#)

Sequence of select operator.

SelOrderSequence

is of datatype [SelOrder](#)

The sequence of order criteria. The first criteria is the importance criteria.

SelValueSequence

is of datatype [SelValue](#)

Sequence of attribute search conditions.

SMatLinkSequence

is of datatype [SMatLink](#)

Sequence of SMatLink objects.

SS_BOOLEAN

is of datatype [S_BOOLEAN](#)

Sequence of boolean sequence.

SS_BYTE

is of datatype [S_BYTE](#)

Sequence of octet sequence.

SS_BYTESTR

is of datatype [S_BYTESTR](#)

Sequence of bytestream sequence.

SS_COMPLEX

is of datatype [S_COMPLEX](#)

Sequence of float complex sequence.

SS_ DATE

is of datatype [S_ DATE](#)

Sequence of date sequence.

SS_ DCOMPLEX

is of datatype [S_ DCOMPLEX](#)

Sequence of double complex sequence.

SS_ DOUBLE

is of datatype [S_ DOUBLE](#)

Sequence of double sequence.

SS_ ExternalReference

is of datatype [S_ ExternalReference](#)

Sequence of external reference sequence.

SS_ FLOAT

is of datatype [S_ FLOAT](#)

Sequence of float sequence.

SS_ LONG

is of datatype [S_ LONG](#)

Sequence of long sequence.

SS_ LONGLONG

is of datatype [S_ LONGLONG](#)

Sequence of longlong sequence.

SS_ SHORT

is of datatype [S_ SHORT](#)

Sequence of short sequence.



SS_STRING

is of datatype [S_STRING](#)

Sequence of sting sequence.

SubMatrixSequence

is of datatype [SubMatrix](#)

Sequence of SubMatrix objects.

T_BYTESTR

is of datatype [T_BYTE](#)

Sequence of byte values. This type is represented in the datatype enumeration by DT_BYTESTR.

Chapter 10

ASAM ODS Exceptions

10.1 AoException

The ASAM ODS API knows only one kind of exception the **AoException**. The different errors are given the field **errCode** of the exception, the explanation of the errors are given in **Definitions of the ErrorCode enumeration**(p. 325). The severity of the exception is given in the field **sevFlag**, the explanation of the severity flag is given in **Definitions of the SeverityFlag enumeration**(p. 335). The minor code is an integer and given in the field **minorCode**. The **minorCode** is the Athos error code, for a detail description of the error look at the technical reference sheet. The reason of the error is given in the field **reason**. The **reason** is the error code as a text and the description and parameter of the error from Athos.

ErrorCode errCode

The code of the detected error.

SeverityFlag sevFlag

The severity of the error.

T_LONG minorCode

The minor code for a more detailed error analysis.

T_STRING reason

The reason of the error.



Chapter 11

ASAM ODS Interface Short Reference

Interface: [AoFactory](#)

- T_STRING [getDescription](#) ();
- T_STRING [getInterfaceVersion](#) ();
- T_STRING [getName](#) ();
- T_STRING [getType](#) ();
- AoSession [newSession](#) (T_STRING auth);

Interface: [AoSession](#)

- void [abortTransaction](#) ();
- void [close](#) ();
- void [commitTransaction](#) ();
- Blob [createBlob](#) ();
- AoSession [createCoSession](#) ();
- QueryEvaluator [createQueryEvaluator](#) ();
- void [flush](#) ();
- ApplElemAccess [getApplElemAccess](#) ();
- ApplicationStructure [getApplicationStructure](#) ();
- ApplicationStructureValue [getApplicationStructureValue](#) ();
- BaseStructure [getBaseStructure](#) ();
- NameValueIterator [getContext](#) (Pattern varPattern);
- NameValue [getContextByName](#) (Name varName);
- T_STRING [getDescription](#) ();

- T_SHORT [getLockMode](#) ();
- Name [getName](#) ();
- T_STRING [getType](#) ();
- InstanceElement [getUser](#) ();
- 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](#) (T_SHORT lockMode);
- void [setPassword](#) (T_STRING username, T_STRING oldPassword, T_STRING newPassword);
- void [startTransaction](#) ();

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 iid);
- ACLSequence [getInstanceRights](#) (T_LONGLONG aid, T_LONGLONG iid);
- 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);

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 aaDataType);
- void [setEnumerationDefinition](#) (EnumerationDefinition enumDef);
- void [setIsAutogenerated](#) (T_BOOLEAN isAutogenerated);
- void [setIsObligatory](#) (T_BOOLEAN aaIsObligatory);
- void [setIsUnique](#) (T_BOOLEAN aaIsUnique);
- void [setLength](#) (T_LONG aaLength);
- void [setName](#) (Name aaName);
- void [setRights](#) (InstanceElement usergroup, T_LONG rights, RightsSet set);
- void [setUnit](#) (T_LONGLONG aaUnit);
- void [withUnit](#) (T_BOOLEAN withUnit);
- void [withValueFlag](#) (T_BOOLEAN withValueFlag);

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 [getRelationsByBaseName](#) (Name baseRelName);
- ApplicationRelationSequence [getRelationsByType](#) (RelationType aeRelationType);
- ACLSequence [getRights](#) ();
- T_LONG [getSecurityLevel](#) ();
- NameSequence [listAllRelatedElements](#) ();
- NameSequence [listAttributes](#) (Pattern aaPattern);
- NameIterator [listInstances](#) (Pattern aaPattern);
- NameSequence [listRelatedElementsByRelationship](#) (Relationship aeRelationship);
- 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);

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);

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);

Interface: [BaseAttribute](#)

- BaseElement [getBaseElement](#) ();
- DataType [getDataType](#) ();
- EnumerationDefinition [getEnumerationDefinition](#) ();
- Name [getName](#) ();
- T_BOOLEAN [isObligatory](#) ();
- T_BOOLEAN [isUnique](#) ();

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);

Interface: [BaseRelation](#)



- BaseElement [getElem1](#) ();
- BaseElement [getElem2](#) ();
- Name [getInverseRelationName](#) ();
- RelationRange [getInverseRelationRange](#) ();
- Relationship [getInverseRelationship](#) ();
- Name [getRelationName](#) ();
- RelationRange [getRelationRange](#) ();
- Relationship [getRelationship](#) ();
- RelationType [getRelationType](#) ();

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);

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);

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);

Interface: [ElemResultSetExtSeqIterator](#)

- void [destroy](#) ();
- T_LONG [getCount](#) ();
- ElemResultSetExtSequence [nextN](#) (T_LONG how_many);
- ElemResultSetExt [nextOne](#) ();
- void [reset](#) ();

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);

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](#) (Relationship ieRelationship, Pattern iePattern);
- void [removeInstanceAttribute](#) (Name attrName);
- void [removeRelation](#) (ApplicationRelation applRel, InstanceElement instElem_nm);
- void [renameInstanceAttribute](#) (Name oldName, Name 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](#) ();

Interface: [InstanceElementIterator](#)

- void [destroy](#) ();
- T_LONG [getCount](#) ();
- InstanceElementSequence [nextN](#) (T_LONG how_many);

- InstanceElement [nextOne](#) ();
- void [reset](#) ();

Interface: [Measurement](#)

- SMatLink [createSMatLink](#) ();
- SMatLinkSequence [getSMatLinks](#) ();
- ValueMatrix [getValueMatrix](#) ();
- void [removeSMatLink](#) (SMatLink smLink);

Interface: [NameIterator](#)

- void [destroy](#) ();
- T_LONG [getCount](#) ();
- NameSequence [nextN](#) (T_LONG how_many);
- Name [nextOne](#) ();
- void [reset](#) ();

Interface: [NameValueIterator](#)

- void [destroy](#) ();
- T_LONG [getCount](#) ();
- NameValueSequence [nextN](#) (T_LONG how_many);
- NameValue [nextOne](#) ();
- void [reset](#) ();

Interface: [NameValueUnitIdIterator](#)

- void [destroy](#) ();
- T_LONG [getCount](#) ();
- NameValueSeqUnitId [nextN](#) (T_LONG how_many);
- NameValueUnitId [nextOne](#) ();
- void [reset](#) ();

Interface: [NameValueUnitIterator](#)

- void [destroy](#) ();
- T_LONG [getCount](#) ();
- NameValueUnitSequence [nextN](#) (T_LONG how_many);
- NameValueUnit [nextOne](#) ();



- void `reset` ();

Interface: `NameValueUnitSequenceIterator`

- void `destroy` ();
- T_LONG `getCount` ();
- NameValueSeqUnitSequence `nextN` (T_LONG how_many);
- NameValueSeqUnit `nextOne` ();
- void `reset` ();

Interface: `Query`

- void `executeQuery` (NameValueSequence params);
- InstanceElementIterator `getInstances` ();
- QueryEvaluator `getQueryEvaluator` ();
- QueryStatus `getStatus` ();
- NameValueSeqUnitSequence `getTable` ();
- NameValueUnitSequenceIterator `getTableRows` ();
- void `prepareQuery` (NameValueSequence params);

Interface: `QueryEvaluator`

- 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);

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](#) (ColumnSequence columns);

Interface: [SubMatrix](#)

- ColumnSequence [getColumns](#) (Pattern colPattern);
- ValueMatrix [getValueMatrix](#) ();
- NameSequence [listColumns](#) (Pattern colPattern);

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, NameValueSeqUnitSequence value);
- void [setValueVector](#) (Column col, SetType set, T_LONG startPoint, TS_ValueSeq value);

Index

- abortTransaction
 - AoSession, [36](#)
- ACL, [297](#)
- ACLSequence, [343](#)
- ADD_RIGHT, [332](#)
- addColumn
 - ValueMatrix, [280](#)
- addColumnScaledBy
 - ValueMatrix, [281](#)
- addInstanceAttribute
 - InstanceElement, [209](#)
- addItem
 - EnumerationDefinition, [205](#)
- AggrFunc, [321](#)
- AIDName, [297](#)
- AIDNameSequence, [343](#)
- AIDNameUnitId, [297](#)
- AIDNameUnitIdSequence, [343](#)
- AIDNameValueSeqUnitId, [298](#)
- AIDNameValueSeqUnitIdSequence, [344](#)
- AIDNameValueUnitId, [298](#)
- ALL, [322](#)
- ALL_REL, [331](#)
- AND, [334](#)
- AO_ACCESS_DENIED, [325](#)
- AO_BAD_OPERATION, [325](#)
- AO_BAD_PARAMETER, [325](#)
- AO_CONNECT_FAILED, [325](#)
- AO_CONNECT_REFUSED, [325](#)
- AO_CONNECTION_LOST, [325](#)
- AO_DUPLICATE_BASE_ATTRIBUTE, [325](#)
- AO_DUPLICATE_NAME, [325](#)
- AO_DUPLICATE_VALUE, [325](#)
- AO_HAS_BASE_ATTRIBUTE, [330](#)
- AO_HAS_BASE_RELATION, [329](#)
- AO_HAS_INSTANCES, [326](#)
- AO_HAS_REFERENCES, [326](#)
- AO_IMPLEMENTATION_PROBLEM, [326](#)
- AO_INCOMPATIBLE_UNITS, [326](#)
- AO_INVALID_ASAM_PATH, [326](#)
- AO_INVALID_ATTRIBUTE_TYPE, [326](#)
- AO_INVALID_BASE_ELEMENT, [326](#)
- AO_INVALID_BASETYPE, [326](#)
- AO_INVALID_BUILDUP_FUNCTION, [327](#)
- AO_INVALID_COLUMN, [327](#)
- AO_INVALID_COUNT, [327](#)
- AO_INVALID_DATATYPE, [327](#)
- AO_INVALID_ELEMENT, [327](#)
- AO_INVALID_LENGTH, [327](#)
- AO_INVALID_ORDINALNUMBER, [327](#)
- AO_INVALID_RELATION, [327](#)
- AO_INVALID_RELATION_RANGE, [327](#)
- AO_INVALID_RELATION_TYPE, [327](#)
- AO_INVALID_RELATIONSHIP, [328](#)
- AO_INVALID_SET_TYPE, [328](#)
- AO_INVALID_SMATLINK, [328](#)
- AO_INVALID_SUBMATRIX, [328](#)
- AO_IS_BASE_ATTRIBUTE, [328](#)
- AO_IS_BASE_RELATION, [328](#)
- AO_IS_MEASUREMENT_MATRIX, [328](#)
- AO_MATH_ERROR, [328](#)
- AO_MISSING_APPLICATION -
ELEMENT, [328](#)
- AO_MISSING_ATTRIBUTE, [328](#)
- AO_MISSING_RELATION, [328](#)
- AO_MISSING_VALUE, [328](#)
- AO_NO_MEMORY, [329](#)
- AO_NO_PATH_TO_ELEMENT, [329](#)
- AO_NO_SCALING_COLUMN, [330](#)
- AO_NOT_FOUND, [329](#)
- AO_NOT_IMPLEMENTED, [329](#)
- AO_NOT_UNIQUE, [329](#)
- AO_OPEN_MODE_NOT_SUPPORTED, [329](#)
- AO_QUERY_INCOMPLETE, [330](#)
- AO_QUERY_INVALID, [330](#)
- AO_QUERY_INVALID_RESULTTYPE, [330](#)
- AO_QUERY_PROCESSING_ERROR, [330](#)
- AO_QUERY_TIMEOUT_EXCEEDED, [330](#)
- AO_QUERY_TYPE_INVALID, [330](#)
- AO_SESSION_LIMIT_REACHED, [329](#)
- AO_SESSION_NOT_ACTIVE, [329](#)
- AO_TRANSACTION_ALREADY -
ACTIVE, [329](#)

- AO_TRANSACTION_NOT_ACTIVE, 329
- AO_UNKNOWN_ERROR, 325
- AO_UNKNOWN_UNIT, 330
- AoException, 18, 353
- AoFactory, 23
 - getDescription, 29
 - getInterfaceVersion, 29
 - getName, 30
 - getType, 30
 - newSession, 31
- AoFactory, 12, 19, 23
- AoFactory.getDescription, 29
- AoFactory.getInterfaceVersion, 29
- AoFactory.getName, 30
- AoFactory.getType, 31
- AoFactory.newSession, 31
- AoService, 13, 21
 - listServices, 13, 22
 - newFactory, 14, 22
- AoService, 12, 13, 19, 21
- AoService.listServices, 13, 22
- AoService.newFactory, 14, 22
- AoServiceFactory, 12, 21
 - newService, 13, 21
- AoServiceFactory, 12, 19, 21
- AoServiceFactory.newService, 13, 21
- AoSession, 32
 - abortTransaction, 36
 - close, 37
 - commitTransaction, 38
 - createBlob, 39
 - createCoSession, 39
 - createQueryEvaluator, 40
 - flush, 41
 - getApplElemAccess, 42
 - getApplicationStructure, 43
 - getApplicationStructureValue, 44
 - getBaseStructure, 44
 - getContext, 45
 - getContextByName, 46
 - getDescription, 47
 - getLockMode, 48
 - getName, 49
 - getType, 49
 - getUser, 50
 - listContext, 50
 - removeContext, 51
 - setContext, 52
 - setContextString, 53
 - setCurrentInitialRights, 55
 - setLockMode, 55
 - setPassword, 56
 - startTransaction, 57
- AoSession, 32
 - abortTransaction, 36
 - close, 37
 - commitTransaction, 38
 - createBlob, 39
 - createCoSession, 39
 - createQueryEvaluator, 40
 - flush, 41
 - getApplElemAccess, 42
 - getApplicationStructure, 43
 - getApplicationStructureValue, 44
 - getBaseStructure, 44
 - getContext, 45
 - getContextByName, 46
 - getDescription, 47
 - getLockMode, 48
 - getName, 49
 - getType, 49
 - getUser, 50
 - listContext, 50
 - removeContext, 51
 - setContext, 52
 - setContextString, 53
 - setCurrentInitialRights, 55
 - setLockMode, 55
 - setPassword, 56
 - startTransaction, 57
- APPEND, 334
- append
 - Blob, 192
- ApplAttr, 299
- APPLATTR_ONLY, 322
- ApplAttrSequence, 344
- ApplElem, 300
- ApplElemAccess, 58
 - deleteInstances, 69
 - getAttributeRights, 70
 - getElementInitialRights, 71
 - getElementRights, 72
 - getInitialRightReference, 73
 - getInstanceInitialRights, 74
 - getInstanceRights, 75
 - getInstances, 77
 - getInstancesExt, 80
 - getRelInst, 82
 - getValueMatrix, 83
 - insertInstances, 83
 - setAttributeRights, 87
 - setElementInitialRights, 88
 - setElementRights, 89
 - setInitialRightReference, 90
 - setInstanceInitialRights, 90
 - setInstanceRights, 91
 - setRelInst, 92

- updateInstances, 93
- ApplElemAccess, 58
- ApplElemAccess.deleteInstances, 69
- ApplElemAccess.getAttributeRights, 70
- ApplElemAccess.getElementInitialRights, 71
- ApplElemAccess.getElementRights, 72
- ApplElemAccess.getInitialRightReference, 73
- ApplElemAccess.getInstanceInitialRights, 74
- ApplElemAccess.getInstanceRights, 76
- ApplElemAccess.getInstances, 77
- ApplElemAccess.getInstancesExt, 81
- ApplElemAccess.getRelInst, 82
- ApplElemAccess.getValueMatrix, 83
- ApplElemAccess.insertInstances, 84
- ApplElemAccess.setAttributeRights, 88
- ApplElemAccess.setElementInitialRights, 88
- ApplElemAccess.setElementRights, 89
- ApplElemAccess.setInitialRightReference, 90
- ApplElemAccess.setInstanceInitialRights, 91
- ApplElemAccess.setInstanceRights, 91
- ApplElemAccess.setRelInst, 92
- ApplElemAccess.updateInstances, 93
- ApplElemSequence, 344
- ApplicationAttribute, 98
 - getApplicationElement, 98
 - getBaseAttribute, 98
 - getDataType, 99
 - getEnumerationDefinition, 99
 - getLength, 100
 - getName, 101
 - getRights, 102
 - getUnit, 103
 - hasUnit, 103
 - hasValueFlag, 104
 - isAutogenerated, 105
 - isObligatory, 105
 - isUnique, 106
 - setBaseAttribute, 106
 - setDataType, 107
 - setEnumerationDefinition, 108
 - setIsAutogenerated, 109
 - setIsObligatory, 109
 - setIsUnique, 110
 - setLength, 111
 - setName, 112
 - setRights, 113
 - setUnit, 114
 - withUnit, 115
 - withValueFlag, 116
- ApplicationAttribute.getApplicationElement, 98
- ApplicationAttribute.getBaseAttribute, 98
- ApplicationAttribute.getDataType, 99
- ApplicationAttribute.getEnumerationDefinition, 99
- ApplicationAttribute.getLength, 100
- ApplicationAttribute.getName, 101
- ApplicationAttribute.getRights, 102
- ApplicationAttribute.getUnit, 103
- ApplicationAttribute.hasUnit, 104
- ApplicationAttribute.hasValueFlag, 104
- ApplicationAttribute.isAutogenerated, 105
- ApplicationAttribute.isObligatory, 105
- ApplicationAttribute.isUnique, 106
- ApplicationAttribute.setBaseAttribute, 106
- ApplicationAttribute.setDataType, 107
- ApplicationAttribute.setEnumerationDefinition, 108
- ApplicationAttribute.setIsAutogenerated, 109
- ApplicationAttribute.setIsObligatory, 109
- ApplicationAttribute.setIsUnique, 110
- ApplicationAttribute.setLength, 111
- ApplicationAttribute.setName, 112
- ApplicationAttribute.setRights, 113
- ApplicationAttribute.setUnit, 114
- ApplicationAttribute.withUnit, 115
- ApplicationAttribute.withValueFlag, 116
- ApplicationAttributeSequence, 344
- ApplicationElement, 117
 - createAttribute, 117
 - createInstance, 118
 - createInstances, 118
 - getAllRelatedElements, 119
 - getAllRelations, 120
 - getApplicationStructure, 120
 - getAttributeByBaseName, 121
 - getAttributeByName, 122
 - getAttributes, 122
 - getBaseElement, 123
 - getId, 124
 - getInitialRightRelations, 124
 - getInitialRights, 125
 - getInstanceById, 126
 - getInstanceByName, 126
 - getInstances, 127
 - getName, 128
 - getRelatedElementsByRelationship, 129
 - getRelationsByBaseName, 129
 - getRelationsByType, 130
 - getRights, 131
 - getSecurityLevel, 131
 - listAllRelatedElements, 132
 - listAttributes, 133
 - listInstances, 133

- listRelatedElementsByRelationship, 134
- removeAttribute, 134
- removeInstance, 135
- setBaseElement, 136
- setInitialRightRelation, 136
- setInitialRights, 137
- setName, 138
- setRights, 139
- setSecurityLevel, 140
- ApplicationElement, 117
- ApplicationElement.createAttribute, 117
- ApplicationElement.createInstance, 118
- ApplicationElement.createInstances, 119
- ApplicationElement.getAllRelatedElements, 119
- ApplicationElement.getAllRelations, 120
- ApplicationElement.getApplicationStructure, 120
- ApplicationElement.getAttributeByBaseName, 121
- ApplicationElement.getAttributeByName, 122
- ApplicationElement.getAttributes, 122
- ApplicationElement.getBaseElement, 123
- ApplicationElement.getId, 124
- ApplicationElement.getInitialRightRelations, 124
- ApplicationElement.getInitialRights, 125
- ApplicationElement.getInstanceById, 126
- ApplicationElement.getInstanceByName, 127
- ApplicationElement.getInstances, 127
- ApplicationElement.getName, 128
- ApplicationElement.getRelatedElementsByRelationship, 129
- ApplicationElement.getRelationsByBaseName, 129
- ApplicationElement.getRelationsByType, 130
- ApplicationElement.getRights, 131
- ApplicationElement.getSecurityLevel, 132
- ApplicationElement.listAllRelatedElements, 132
- ApplicationElement.listAttributes, 133
- ApplicationElement.listInstances, 133
- ApplicationElement.listRelatedElementsByRelationship, 134
- ApplicationElement.removeAttribute, 134
- ApplicationElement.removeInstance, 135
- ApplicationElement.setBaseElement, 136
- ApplicationElement.setInitialRightRelation, 137
- ApplicationElement.setInitialRights, 137
- ApplicationElement.setName, 138
- ApplicationElement.setRights, 139
- ApplicationElement.setSecurityLevel, 140
- ApplicationElementSequence, 344
- ApplicationRelation, 141
 - getBaseRelation, 144
 - getElem1, 145
 - getElem2, 145
 - getInverseRelationName, 146
 - getInverseRelationRange, 146
 - getInverseRelationship, 147
 - getRelationName, 147
 - getRelationRange, 148
 - getRelationship, 148
 - getRelationType, 149
 - setBaseRelation, 149
 - setElem1, 150
 - setElem2, 151
 - setInverseRelationName, 151
 - setInverseRelationRange, 152
 - setRelationName, 153
 - setRelationRange, 153
 - setRelationType, 154
- ApplicationRelation, 141
 - getBaseRelation, 144
 - getElem1, 145
 - getElem2, 145
 - getInverseRelationName, 146
 - getInverseRelationRange, 147
 - getInverseRelationship, 147
 - getRelationName, 147
 - getRelationRange, 148
 - getRelationship, 149
 - getRelationType, 149
 - setBaseRelation, 149
 - setElem1, 150
 - setElem2, 151
 - setInverseRelationName, 151
 - setInverseRelationRange, 152
 - setRelationName, 153
 - setRelationRange, 153
 - setRelationType, 154
- ApplicationRelationInstanceElementSeq, 301
- ApplicationRelationInstanceElementSeqSequence, 344
- ApplicationRelationSequence, 344
- ApplicationStructure, 155
 - check, 155
 - createElement, 156

- createEnumerationDefinition, 157
- createInstanceRelations, 158
- createRelation, 159
- getElementById, 160
- getElementByName, 161
- getElements, 162
- getElementsByBaseType, 163
- getEnumerationDefinition, 164
- getInstanceByAsamPath, 165
- getInstancesById, 165
- getRelations, 166
- getSession, 167
- getTopLevelElements, 167
- listElements, 168
- listElementsByBaseType, 169
- listEnumerations, 170
- listTopLevelElements, 170
- removeElement, 171
- removeEnumerationDefinition, 172
- removeRelation, 173
- ApplicationStructure, 155
- ApplicationStructure.check, 155
- ApplicationStructure.createElement, 156
- ApplicationStructure.createEnumerationDefinition, 157
- ApplicationStructure.createInstanceRelations, 158
- ApplicationStructure.createRelation, 159
- ApplicationStructure.getElementById, 160
- ApplicationStructure.getElementByName, 161
- ApplicationStructure.getElements, 162
- ApplicationStructure.getElementsByBaseType, 163
- ApplicationStructure.getEnumerationDefinition, 164
- ApplicationStructure.getInstanceByAsamPath, 165
- ApplicationStructure.getInstancesById, 166
- ApplicationStructure.getRelations, 166
- ApplicationStructure.getSession, 167
- ApplicationStructure.getTopLevelElements, 167
- ApplicationStructure.listElements, 168
- ApplicationStructure.listElementsByBaseType, 169
- ApplicationStructure.listEnumerations, 170
- ApplicationStructure.listTopLevelElements, 170
- ApplicationStructure.removeElement, 171
- ApplicationStructure.removeEnumerationDefinition, 172
- ApplicationStructure.removeRelation, 173
- ApplicationStructureValue, 301
- ApplRel, 301
- ApplRelSequence, 344
- ASAM ODS Exceptions, 353
- ASAM ODS Types, 341
- AsamOdsConstants, 337
- AsamOdsEnumerations, 321
- AsamOdsStructures, 297
- AsamOdsUnions, 315
- ATF File, 19
- ATF/CLA, 19
- ATF/XML, 19
- AttrResultSet, 303
- AttrResultSetSequence, 345
- AttrType, 321
- AVG, 321
- BaseAttribute, 174
 - getBaseElement, 174
 - getDataType, 175
 - getEnumerationDefinition, 175
 - getName, 175
 - isObligatory, 176
 - isUnique, 176
- BaseAttribute, 174
- BaseAttribute.getBaseElement, 174
- BaseAttribute.getDataType, 175
- BaseAttribute.getEnumerationDefinition, 175
- BaseAttribute.getName, 176
- BaseAttribute.isObligatory, 176
- BaseAttribute.isUnique, 177
- BaseAttributeSequence, 345
- BaseElement, 177
 - getAllRelations, 177
 - getAttributes, 178
 - getRelatedElementsByRelationship, 178
 - getRelationsByType, 179
 - getType, 179
 - isTopLevel, 180
 - listAttributes, 180
 - listRelatedElementsByRelationship, 181
- BaseElement, 177
- BaseElement.getAllRelations, 177
- BaseElement.getAttributes, 178
- BaseElement.getRelatedElementsByRelationship, 178
- BaseElement.getRelationsByType, 179
- BaseElement.getType, 179
- BaseElement.isTopLevel, 180
- BaseElement.listAttributes, 180
- BaseElement.listRelatedElementsByRelationship, 181
- BaseElementSequence, 345
- BaseRelation, 181

- getElem1, 181
- getElem2, 182
- getInverseRelationName, 182
- getInverseRelationRange, 183
- getInverseRelationship, 183
- getRelationName, 184
- getRelationRange, 184
- getRelationship, 184
- getRelationType, 185
- BaseRelation, 181
- BaseRelation.getElem1, 181
- BaseRelation.getElem2, 182
- BaseRelation.getInverseRelationName, 182
- BaseRelation.getInverseRelationRange, 183
- BaseRelation.getInverseRelationship, 183
- BaseRelation.getRelationName, 184
- BaseRelation.getRelationRange, 184
- BaseRelation.getRelationship, 185
- BaseRelation.getRelationType, 185
- BaseRelationSequence, 345
- BaseStructure, 185
 - getElementByType, 185
 - getElements, 186
 - getRelation, 187
 - getTopLevelElements, 188
 - getVersion, 189
 - listElements, 190
 - listTopLevelElements, 191
- BaseStructure, 185
- BaseStructure.getElementByType, 186
- BaseStructure.getElements, 187
- BaseStructure.getRelation, 187
- BaseStructure.getTopLevelElements, 188
- BaseStructure.getVersion, 189
- BaseStructure.listElements, 190
- BaseStructure.listTopLevelElements, 191
- BaseType, 342
- BaseTypeSequence, 345
- Blob, 192
 - append, 192
 - compare, 192
 - destroy, 193
 - get, 193
 - getHeader, 194
 - getLength, 194
 - set, 195
 - setHeader, 195
- Blob, 192
- Blob.append, 192
- Blob.compare, 193
- Blob.destroy, 193
- Blob.get, 194
- Blob.getHeader, 194
- Blob.getLength, 195
- Blob.set, 195
- Blob.setHeader, 196
- boolean, 341
- BuildUpFunction, 322
- BUP_JOIN, 322
- BUP_MERGE, 322
- BUP_SORT, 322
- check
 - ApplicationStructure, 155
- CHILD, 331
- CI_EQ, 333
- CI_GT, 333
- CI_GTE, 333
- CI_INSET, 333
- CI_LIKE, 333
- CI_LT, 333
- CI_LTE, 333
- CI_NEQ, 333
- CI_NOTINSET, 333
- CLOSE, 334
- close
 - AoSession, 37
- Column, 196
 - destroy, 196
 - getDataType, 197
 - getFormula, 198
 - getName, 198
 - getSourceMQ, 199
 - getUnit, 199
 - isIndependent, 200
 - isScaling, 200
 - setFormula, 201
 - setIndependent, 201
 - setScaling, 202
 - setUnit, 202
- Column, 196
- Column.destroy, 196
- Column.getDataType, 197
- Column.getFormula, 198
- Column.getName, 198
- Column.getSourceMQ, 199
- Column.getUnit, 199
- Column.isIndependent, 200
- Column.isScaling, 200
- Column.setFormula, 201
- Column.setIndependent, 201
- Column.setScaling, 202
- Column.setUnit, 202
- ColumnSequence, 345
- commitTransaction
 - AoSession, 38
- compare
 - Blob, 192

- InstanceElement, 210
- COMPLETE, 331
- CORBA-Naming Service, 19
- COUNT, 321
- createAttribute
 - ApplicationElement, 117
- createBlob
 - AoSession, 39
- createCoSession
 - AoSession, 39
- createElement
 - ApplicationStructure, 156
- createEnumerationDefinition
 - ApplicationStructure, 157
- createInstance
 - ApplicationElement, 118
- createInstanceRelations
 - ApplicationStructure, 158
- createInstances
 - ApplicationElement, 118
- createQuery
 - QueryEvaluator, 258
- createQueryEvaluator
 - AoSession, 40
- createRelatedInstances
 - InstanceElement, 210
- createRelation
 - ApplicationStructure, 159
 - InstanceElement, 211
- createSMatLink
 - Measurement, 237
- DataType, 322
- DCOUNT, 321
- DEBUGLEVEL, 12
- deepCopy
 - InstanceElement, 212
- deleteInstances
 - AppElemAccess, 69
- destroy
 - Blob, 193
 - Column, 196
 - ElemResultSetExtSeqIterator, 203
 - InstanceElement, 213
 - InstanceElementIterator, 234
 - NameIterator, 240
 - NameValueIterator, 243
 - NameValueUnitIdIterator, 245
 - NameValueUnitIterator, 249
 - NameValueUnitSequenceIterator, 252
 - ValueMatrix, 281
- destroy, 4
- double, 341
- DS_BOOLEAN, 324
- DS_BYTE, 324
- DS_BYTESTR, 324
- DS_COMPLEX, 324
- DS_DATE, 324
- DS_DCOMPLEX, 324
- DS_DOUBLE, 324
- DS_ENUM, 324
- DS_EXTERNALREFERENCE, 324
- DS_FLOAT, 324
- DS_ID, 324
- DS_LONG, 324
- DS_LONGLONG, 324
- DS_SHORT, 323
- DS_STRING, 323
- DT_BLOB, 323
- DT_BOOLEAN, 323
- DT_BYTE, 323
- DT_BYTESTR, 323
- DT_COMPLEX, 323
- DT_DATE, 323
- DT_DCOMPLEX, 323
- DT_DOUBLE, 323
- DT_ENUM, 324
- DT_EXTERNALREFERENCE, 324
- DT_FLOAT, 322
- DT_ID, 323
- DT_LONG, 323
- DT_LONGLONG, 323
- DT_SHORT, 322
- DT_STRING, 322
- DT_UNKNOWN, 322
- ElemId, 303
- ElemIdSequence, 345
- ElemResultSet, 303
- ElemResultSetExt, 304
- ElemResultSetExtSeqIterator, 203
 - destroy, 203
 - getCount, 203
 - nextN, 204
 - nextOne, 204
 - reset, 204
- ElemResultSetExtSeqIterator, 203
- ElemResultSetExtSeqIterator.destroy, 203
- ElemResultSetExtSeqIterator.getCount, 203
- ElemResultSetExtSeqIterator.nextN, 204
- ElemResultSetExtSeqIterator.nextOne, 204
- ElemResultSetExtSeqIterator.reset, 205
- ElemResultSetExtSequence, 345
- ElemResultSetSequence, 345
- EnumerationDefinition, 205
 - addItem, 205
 - getIndex, 206
 - getItem, 206

- getItemName, 207
- getName, 207
- listItemNames, 208
- renameItem, 208
- setName, 209
- EnumerationDefinition, 205
- EnumerationDefinition.addItem, 205
- EnumerationDefinition.getIndex, 206
- EnumerationDefinition.getItem, 206
- EnumerationDefinition.getItemName, 207
- EnumerationDefinition.getName, 207
- EnumerationDefinition.listItemNames, 208
- EnumerationDefinition.renameItem, 208
- EnumerationDefinition.setName, 209
- EQ, 332
- ERROR, 335
- ErrorCode, 325
- examples, 18
- executeQuery
 - Query, 254
- FATHER, 331
- FATHER_CHILD, 332
- float, 341
- flush
 - AoSession, 41
- get
 - Blob, 193
- getAllRelatedElements
 - ApplicationElement, 119
- getAllRelations
 - ApplicationElement, 120
 - BaseElement, 177
- getApplElemAccess
 - AoSession, 42
- getApplicationElement
 - ApplicationAttribute, 98
 - InstanceElement, 214
- getApplicationStructure
 - AoSession, 43
 - ApplicationElement, 120
- getApplicationStructureValue
 - AoSession, 44
- getAsamPath
 - InstanceElement, 215
- getAttributeByBaseName
 - ApplicationElement, 121
- getAttributeByName
 - ApplicationElement, 122
- getAttributeRights
 - ApplElemAccess, 70
- getAttributes
 - ApplicationElement, 122
- BaseElement, 178
- getBaseAttribute
 - ApplicationAttribute, 98
- getBaseElement
 - ApplicationElement, 123
 - BaseAttribute, 174
- getBaseRelation
 - ApplicationRelation, 144
- getBaseStructure
 - AoSession, 44
- getColumnCount
 - ValueMatrix, 282
- getColumns
 - SubMatrix, 268
 - ValueMatrix, 283
- getColumnsScaledBy
 - ValueMatrix, 284
- getContext
 - AoSession, 45
- getContextByName
 - AoSession, 46
- getCount
 - ElemResultSetExtSeqIterator, 203
 - InstanceElementIterator, 235
 - NameIterator, 240
 - NameValueIterator, 243
 - NameValueUnitIdIterator, 246
 - NameValueUnitIterator, 250
 - NameValueUnitSequenceIterator, 252
- getDataType
 - ApplicationAttribute, 99
 - BaseAttribute, 175
 - Column, 197
- getDescription
 - AoFactory, 29
 - AoSession, 47
- getElem1
 - ApplicationRelation, 145
 - BaseRelation, 181
- getElem2
 - ApplicationRelation, 145
 - BaseRelation, 182
- getElementById
 - ApplicationStructure, 160
- getElementByName
 - ApplicationStructure, 161
- getElementByType
 - BaseStructure, 185
- getElementInitialRights
 - ApplElemAccess, 71
- getElementRights
 - ApplElemAccess, 72
- getElements
 - ApplicationStructure, 162

- BaseStructure, 186
- getElementsByBaseType
 - ApplicationStructure, 163
- getEnumerationDefinition
 - ApplicationAttribute, 99
 - ApplicationStructure, 164
 - BaseAttribute, 175
- getFormula
 - Column, 198
- getHeader
 - Blob, 194
- getId
 - ApplicationElement, 124
 - InstanceElement, 215
- getIndependentColumns
 - ValueMatrix, 284
- getIndex
 - EnumerationDefinition, 206
- getInitialRightReference
 - AppElemAccess, 73
- getInitialRightRelations
 - ApplicationElement, 124
- getInitialRights
 - ApplicationElement, 125
 - InstanceElement, 217
- getInstanceByAsamPath
 - ApplicationStructure, 165
- getInstanceById
 - ApplicationElement, 126
- getInstanceByName
 - ApplicationElement, 126
- getInstanceInitialRights
 - AppElemAccess, 74
- getInstanceRights
 - AppElemAccess, 75
- getInstances
 - AppElemAccess, 77
 - ApplicationElement, 127
 - Query, 255
 - QueryEvaluator, 259
- getInstancesById
 - ApplicationStructure, 165
- getInstancesExt
 - AppElemAccess, 80
- getInterfaceVersion
 - AoFactory, 29
- getInverseRelationName
 - ApplicationRelation, 146
 - BaseRelation, 182
- getInverseRelationRange
 - ApplicationRelation, 146
 - BaseRelation, 183
- getInverseRelationship
 - ApplicationRelation, 147
 - BaseRelation, 183
- getItem
 - EnumerationDefinition, 206
- getItemName
 - EnumerationDefinition, 207
- getLength
 - ApplicationAttribute, 100
 - Blob, 194
- getLinkType
 - SMatLink, 262
- getLockMode
 - AoSession, 48
- getName
 - AoFactory, 30
 - AoSession, 49
 - ApplicationAttribute, 101
 - ApplicationElement, 128
 - BaseAttribute, 175
 - Column, 198
 - EnumerationDefinition, 207
 - InstanceElement, 218
- getOrdinalNumber
 - SMatLink, 262
- getQueryEvaluator
 - Query, 255
- getRelatedElementsByRelationship
 - ApplicationElement, 129
 - BaseElement, 178
- getRelatedInstances
 - InstanceElement, 219
- getRelatedInstancesByRelationship
 - InstanceElement, 220
- getRelation
 - BaseStructure, 187
- getRelationName
 - ApplicationRelation, 147
 - BaseRelation, 184
- getRelationRange
 - ApplicationRelation, 148
 - BaseRelation, 184
- getRelations
 - ApplicationStructure, 166
- getRelationsByBaseName
 - ApplicationElement, 129
- getRelationsByType
 - ApplicationElement, 130
 - BaseElement, 179
- getRelationship
 - ApplicationRelation, 148
 - BaseRelation, 184
- getRelationType
 - ApplicationRelation, 149
 - BaseRelation, 185
- getRelInst

- ApplElemAccess, 82
- getRights
 - ApplicationAttribute, 102
 - ApplicationElement, 131
 - InstanceElement, 221
- getRowCount
 - ValueMatrix, 285
- getScalingColumns
 - ValueMatrix, 285
- getSecurityLevel
 - ApplicationElement, 131
- getSession
 - ApplicationStructure, 167
- getSMat1
 - SMatLink, 262
- getSMat1Columns
 - SMatLink, 263
- getSMat2
 - SMatLink, 263
- getSMat2Columns
 - SMatLink, 264
- getSMatLinks
 - Measurement, 238
- getSourceMQ
 - Column, 199
- getStatus
 - Query, 256
- getTable
 - Query, 256
 - QueryEvaluator, 260
- getTableRows
 - Query, 257
 - QueryEvaluator, 261
- getTopLevelElements
 - ApplicationStructure, 167
 - BaseStructure, 188
- getType
 - AoFactory, 30
 - AoSession, 49
 - BaseElement, 179
- getUnit
 - ApplicationAttribute, 103
 - Column, 199
- getUser
 - AoSession, 50
- getValue
 - InstanceElement, 222
 - ValueMatrix, 286
- getValueByBaseName
 - InstanceElement, 223
- getValueInUnit
 - InstanceElement, 223
- getValueMatrix
 - ApplElemAccess, 83
 - Measurement, 238
 - SubMatrix, 268
- getValueMeaPoint
 - ValueMatrix, 287
- getValueSeq
 - InstanceElement, 224
- getValueVector
 - ValueMatrix, 287
- getVersion
 - BaseStructure, 189
- GT, 333
- GTE, 333
- hasUnit
 - ApplicationAttribute, 103
- hasValueFlag
 - ApplicationAttribute, 104
- INCOMPLETE, 331
- INFO, 332
- INFO_FROM, 331
- INFO_REL, 331
- INFO_TO, 331
- INFORMATION, 335
- INHERITANCE, 332
- InitialRight, 304
- InitialRightSequence, 346
- INSERT, 334
- insertInstances
 - ApplElemAccess, 83
- INSET, 333
- InstanceElement, 209
 - addInstanceAttribute, 209
 - compare, 210
 - createRelatedInstances, 210
 - createRelation, 211
 - deepCopy, 212
 - destroy, 213
 - getApplicationElement, 214
 - getAsamPath, 215
 - getId, 215
 - getInitialRights, 217
 - getName, 218
 - getRelatedInstances, 219
 - getRelatedInstancesByRelationship, 220
 - getRights, 221
 - getValue, 222
 - getValueByBaseName, 223
 - getValueInUnit, 223
 - getValueSeq, 224
 - listAttributes, 224
 - listRelatedInstances, 225
 - listRelatedInstancesByRelationship, 226
 - removeInstanceAttribute, 226

- removeRelation, 227
- renameInstanceAttribute, 227
- setInitialRights, 228
- setName, 229
- setRights, 229
- setValue, 230
- setValueSeq, 231
- shallowCopy, 232
- upcastMeasurement, 232
- upcastSubMatrix, 233
- InstanceElement, 209
- InstanceElement.addInstanceAttribute, 209
- InstanceElement.compare, 210
- InstanceElement.createRelatedInstances, 211
- InstanceElement.createRelation, 211
- InstanceElement.deepCopy, 212
- InstanceElement.destroy, 213
- InstanceElement.getApplicationElement, 214
- InstanceElement.getAsamPath, 215
- InstanceElement.getId, 216
- InstanceElement.getInitialRights, 217
- InstanceElement.getName, 218
- InstanceElement.getRelatedInstances, 219
- InstanceElement.getRelatedInstancesByRelationship, 220
- InstanceElement.getRights, 221
- InstanceElement.getValue, 222
- InstanceElement.getValueByBaseName, 223
- InstanceElement.getValueInUnit, 223
- InstanceElement.getValueSeq, 224
- InstanceElement.listAttributes, 225
- InstanceElement.listRelatedInstances, 225
- InstanceElement.listRelatedInstancesByRelationship, 226
- InstanceElement.removeInstanceAttribute, 226
- InstanceElement.removeRelation, 227
- InstanceElement.renameInstanceAttribute, 228
- InstanceElement.setInitialRights, 228
- InstanceElement.setName, 229
- InstanceElement.setRights, 230
- InstanceElement.setValue, 230
- InstanceElement.setValueSeq, 231
- InstanceElement.shallowCopy, 232
- InstanceElement.upcastMeasurement, 233
- InstanceElement.upcastSubMatrix, 233
- InstanceElementIterator, 234
 - destroy, 234
 - getCount, 235
 - nextN, 235
 - nextOne, 236
 - reset, 237
- InstanceElementIterator, 234
- InstanceElementIterator.destroy, 234
- InstanceElementIterator.getCount, 235
- InstanceElementIterator.nextN, 235
- InstanceElementIterator.nextOne, 236
- InstanceElementIterator.reset, 237
- InstanceElementSequence, 346
- INSTATTR_ONLY, 322
- IS_NOT_NULL, 333
- IS_NULL, 333
- isAutogenerated
 - ApplicationAttribute, 105
- isIndependent
 - Column, 200
- isObligatory
 - ApplicationAttribute, 105
 - BaseAttribute, 176
- isScaling
 - Column, 200
- isTopLevel
 - BaseElement, 180
- isUnique
 - ApplicationAttribute, 105
 - BaseAttribute, 176
- JoinDef, 305
- JoinDefSequence, 346
- JoinType, 330
- JTDEFAULT, 330
- JTOUTER, 330
- LIKE, 333
- listAllRelatedElements
 - ApplicationElement, 132
- listAttributes
 - ApplicationElement, 133
 - BaseElement, 180
 - InstanceElement, 224
- listColumns
 - SubMatrix, 269
 - ValueMatrix, 288
- listColumnsScaledBy
 - ValueMatrix, 289
- listContext
 - AoSession, 50
- listElements
 - ApplicationStructure, 168
 - BaseStructure, 190
- listElementsByBaseType
 - ApplicationStructure, 169
- listEnumerations
 - ApplicationStructure, 170
- listIndependentColumns

- ValueMatrix, 289
- listInstances
 - ApplicationElement, 133
- listItemNames
 - EnumerationDefinition, 208
- listRelatedElementsByRelationship
 - ApplicationElement, 134
 - BaseElement, 181
- listRelatedInstances
 - InstanceElement, 225
- listRelatedInstancesByRelationship
 - InstanceElement, 226
- listScalingColumns
 - ValueMatrix, 290
- listServices
 - AoService, 13, 22
- listTopLevelElements
 - ApplicationStructure, 170
 - BaseStructure, 191
- LockMode, 337
- logging, 12
- long, 341
- LT, 332
- LTE, 333
- MAX, 321
- Measurement, 237
 - createSMatLink, 237
 - getSMatLinks, 238
 - getValueMatrix, 238
 - removeSMatLink, 239
- Measurement, 237
- Measurement.createSMatLink, 238
- Measurement.getSMatLinks, 238
- Measurement.getValueMatrix, 239
- Measurement.removeSMatLink, 239
- MIN, 321
- Name, 342
- NameIterator, 240
 - destroy, 240
 - getCount, 240
 - nextN, 241
 - nextOne, 241
 - reset, 242
- NameIterator, 240
- NameIterator.destroy, 240
- NameIterator.getCount, 240
- NameIterator.nextN, 241
- NameIterator.nextOne, 242
- NameIterator.reset, 242
- NameSequence, 346
- NameUnit, 305
- NameValue, 305
- NameValueIterator, 243
 - destroy, 243
 - getCount, 243
 - nextN, 244
 - nextOne, 244
 - reset, 245
- NameValueIterator, 243
- NameValueIterator.destroy, 243
- NameValueIterator.getCount, 243
- NameValueIterator.nextN, 244
- NameValueIterator.nextOne, 244
- NameValueIterator.reset, 245
- NameValueSequence, 346
- NameValueSeqUnit, 306
- NameValueSeqUnitId, 306
- NameValueSeqUnitIdSequence, 346
- NameValueSeqUnitSequence, 346
- NameValueUnit, 307
- NameValueUnitId, 307
- NameValueUnitIdIterator, 245
 - destroy, 245
 - getCount, 246
 - nextN, 246
 - nextOne, 247
 - reset, 247
- NameValueUnitIdIterator, 245
- NameValueUnitIdIterator.destroy, 246
- NameValueUnitIdIterator.getCount, 246
- NameValueUnitIdIterator.nextN, 246
- NameValueUnitIdIterator.nextOne, 247
- NameValueUnitIdIterator.reset, 247
- NameValueUnitIterator, 248
 - destroy, 249
 - getCount, 250
 - nextN, 250
 - nextOne, 251
 - reset, 251
- NameValueUnitIterator, 248
- NameValueUnitIterator.destroy, 249
- NameValueUnitIterator.getCount, 250
- NameValueUnitIterator.nextN, 250
- NameValueUnitIterator.nextOne, 251
- NameValueUnitIterator.reset, 251
- NameValueUnitSequence, 346
- NameValueUnitSequenceIterator, 252
 - destroy, 252
 - getCount, 252
 - nextN, 252
 - nextOne, 253
 - reset, 253
- NameValueUnitSequenceIterator, 252
- NameValueUnitSequenceIterator.destroy, 252

- NameValueUnitSequenceIterator.getCount, 252
- NameValueUnitSequenceIterator.nextN, 253
- NameValueUnitSequenceIterator.nextOne, 253
- NameValueUnitSequenceIterator.reset, 253
- NEQ, 332
- newFactory
 - AoService, 14, 22
- newService
 - AoServiceFactory, 13, 21
- newSession
 - AoFactory, 31
- nextN
 - ElemResultSetExtSeqIterator, 204
 - InstanceElementIterator, 235
 - NameIterator, 241
 - NameValueIterator, 244
 - NameValueUnitIdIterator, 246
 - NameValueUnitIterator, 250
 - NameValueUnitSequenceIterator, 252
- nextOne
 - ElemResultSetExtSeqIterator, 204
 - InstanceElementIterator, 236
 - NameIterator, 241
 - NameValueIterator, 244
 - NameValueUnitIdIterator, 247
 - NameValueUnitIterator, 251
 - NameValueUnitSequenceIterator, 253
- NONE, 321
- NOT, 334
- NOTINSET, 333
- octet, 341
- ODS_LOGFILE, 12
- OPEN, 334
- OR, 334
- Parameter, 17
- Pattern, 342
- POA, 4
- portable object adapter, 4
- prepareQuery
 - Query, 258
- Query, 254
 - executeQuery, 254
 - getInstances, 255
 - getQueryEvaluator, 255
 - getStatus, 256
 - getTable, 256
 - getTableRows, 257
 - prepareQuery, 258
- Query, 254
 - executeQuery, 254
 - getInstances, 255
 - getQueryEvaluator, 255
 - getStatus, 256
 - getTable, 256
 - getTableRows, 257
 - prepareQuery, 258
 - QueryConstants, 337
 - QueryEvaluator, 258
 - createQuery, 258
 - getInstances, 259
 - getTable, 260
 - getTableRows, 261
 - QueryEvaluator, 258
 - QueryEvaluator.createQuery, 259
 - QueryEvaluator.getInstances, 259
 - QueryEvaluator.getTable, 260
 - QueryEvaluator.getTableRows, 261
 - QueryStatus, 331
 - QueryStructure, 307
 - QueryStructureExt, 309
- RelationRange, 143, 310
- Relationship, 142, 331
- RelationType, 141, 332
- REMOVE, 334
- REMOVE_RIGHT, 332
- removeAttribute
 - ApplicationElement, 134
- removeContext
 - AoSession, 51
- removeElement
 - ApplicationStructure, 171
- removeEnumerationDefinition
 - ApplicationStructure, 172
- removeInstance
 - ApplicationElement, 135
- removeInstanceAttribute
 - InstanceElement, 226
- removeRelation
 - ApplicationStructure, 173
 - InstanceElement, 227
- removeSMatLink
 - Measurement, 239
- removeValueMeaPoint
 - ValueMatrix, 290
- removeValueVector
 - ValueMatrix, 291
- renameInstanceAttribute
 - InstanceElement, 227
- renameItem
 - EnumerationDefinition, 208
- reset
 - ElemResultSetExtSeqIterator, 204

- InstanceElementIterator, 237
- NameIterator, 242
- NameValueIterator, 245
- NameValueUnitIdIterator, 247
- NameValueUnitIterator, 251
- NameValueUnitSequenceIterator, 253
- ResultSetExt, 310
- ResultSetExtSequence, 346
- ResultType, 338
- Return, 17
- RightsSet, 332
- S_BLOB, 347
- S_BOOLEAN, 347
- S_BYTE, 347
- S_BYTESTR, 347
- S_COMPLEX, 347
- S_DATE, 347
- S_DCOMPLEX, 347
- S_DOUBLE, 347
- S_ExternalReference, 348
- S_FLOAT, 348
- S_LONG, 348
- S_LONGLONG, 348
- S_SHORT, 348
- S_STRING, 348
- Scalar, 341
- SecurityLevel, 338
- SecurityRights, 339
- SEL_OPERATOR_TYPE, 334
- SEL_VALUE_TYPE, 334
- SelAIDNameUnitId, 310
- SelAIDNameUnitIdSequence, 348
- SelItem, 315
- SelItemSequence, 349
- SelOpcode, 332
- SelOperator, 334
- SelOperatorSequence, 349
- SelOrder, 311
- SelOrderSequence, 349
- SelType, 334
- SelValue, 311
- SelValueExt, 311
- SelValueSequence, 349
- Sequence, 343
- set
 - Blob, 195
- SET_RIGHT, 332
- setAttributeRights
 - ApplElemAccess, 87
- setBaseAttribute
 - ApplicationAttribute, 106
- setBaseElement
 - ApplicationElement, 136
- setBaseRelation
 - ApplicationRelation, 149
- setContext
 - AoSession, 52
- setContextString
 - AoSession, 53
- setCurrentInitialRights
 - AoSession, 55
- setDataType
 - ApplicationAttribute, 107
- setElem1
 - ApplicationRelation, 150
- setElem2
 - ApplicationRelation, 151
- setElementInitialRights
 - ApplElemAccess, 88
- setElementRights
 - ApplElemAccess, 89
- setEnumerationDefinition
 - ApplicationAttribute, 108
- setFormula
 - Column, 201
- setHeader
 - Blob, 195
- setIndependent
 - Column, 201
- setInitialRightReference
 - ApplElemAccess, 90
- setInitialRightRelation
 - ApplicationElement, 136
- setInitialRights
 - ApplicationElement, 137
 - InstanceElement, 228
- setInstanceInitialRights
 - ApplElemAccess, 90
- setInstanceRights
 - ApplElemAccess, 91
- setInverseRelationName
 - ApplicationRelation, 151
- setInverseRelationRange
 - ApplicationRelation, 152
- setIsAutogenerated
 - ApplicationAttribute, 109
- setIsObligatory
 - ApplicationAttribute, 109
- setIsUnique
 - ApplicationAttribute, 110
- setLength
 - ApplicationAttribute, 111
- setLinkType
 - SMatLink, 264
- setLockMode
 - AoSession, 55
- setName

- ApplicationAttribute, 112
- ApplicationElement, 138
- EnumerationDefinition, 209
- InstanceElement, 229
- setOrdinalNumber
 - SMatLink, 265
- setPassword
 - AoSession, 56
- setRelationName
 - ApplicationRelation, 153
- setRelationRange
 - ApplicationRelation, 153
- setRelationType
 - ApplicationRelation, 154
- setRelInst
 - ApplElemAccess, 92
- setRights
 - ApplicationAttribute, 113
 - ApplicationElement, 139
 - InstanceElement, 229
- setScaling
 - Column, 202
- setSecurityLevel
 - ApplicationElement, 140
- setSMat1
 - SMatLink, 265
- setSMat1Columns
 - SMatLink, 266
- setSMat2
 - SMatLink, 266
- setSMat2Columns
 - SMatLink, 267
- SetType, 334
- setUnit
 - ApplicationAttribute, 114
 - Column, 202
- setValue
 - InstanceElement, 230
 - ValueMatrix, 292
- setValueMeaPoint
 - ValueMatrix, 293
- setValueSeq
 - InstanceElement, 231
- setValueVector
 - ValueMatrix, 294
- SeverityFlag, 335
- shallowCopy
 - InstanceElement, 232
- short, 341
- SMatLink, 261
 - getLinkType, 262
 - getOrdinalNumber, 262
 - getSMat1, 262
 - getSMat1Columns, 263
 - getSMat2, 263
 - getSMat2Columns, 264
 - setLinkType, 264
 - setOrdinalNumber, 265
 - setSMat1, 265
 - setSMat1Columns, 266
 - setSMat2, 266
 - setSMat2Columns, 267
- SMatLink, 261
 - SMatLink.getLinkType, 262
 - SMatLink.getOrdinalNumber, 262
 - SMatLink.getSMat1, 263
 - SMatLink.getSMat1Columns, 263
 - SMatLink.getSMat2, 264
 - SMatLink.getSMat2Columns, 264
 - SMatLink.setLinkType, 265
 - SMatLink.setOrdinalNumber, 265
 - SMatLink.setSMat1, 266
 - SMatLink.setSMat1Columns, 266
 - SMatLink.setSMat2, 267
 - SMatLink.setSMat2Columns, 267
- SMatLinkSequence, 349
- SS_BOOLEAN, 349
- SS_BYTE, 349
- SS_BYTESTR, 349
- SS_COMPLEX, 349
- SS_DATE, 350
- SS_DCOMPLEX, 350
- SS_DOUBLE, 350
- SS_ExternalReference, 350
- SS_FLOAT, 350
- SS_LONG, 350
- SS_LOGLONG, 350
- SS_SHORT, 350
- SS_STRING, 350
- startTransaction
 - AoSession, 57
- STDDEV, 321
- string, 341
- SubMatrix, 268
 - getColumns, 268
 - getValueMatrix, 268
 - listColumns, 269
- SubMatrix, 268
 - SubMatrix.getColumns, 268
 - SubMatrix.getValueMatrix, 268
 - SubMatrix.listColumns, 269
- SubMatrixSequence, 351
- SUBTYPE, 331
- SUCCESS, 335
- SUPERTYPE, 331
- T_BLOB, 342
- T_BOOLEAN, 342

- T_BYTE, 342
- T_BYTESTR, 351
- T_COMPLEX, 312
- T_DATE, 342
- T_DCOMPLEX, 312
- T_DOUBLE, 342
- T_ExternalReference, 313
- T_FLOAT, 343
- T_LONG, 343
- T_LOGLONG, 313
- T_SHORT, 343
- T_STRING, 343
- TS_Union, 315
- TS_UnionSeq, 317
- TS_Value, 313
- TS_ValueSeq, 314
- upcastMeasurement
 - InstanceElement, 232
- upcastSubMatrix
 - InstanceElement, 233
- UPDATE, 334
- updateInstances
 - ApplElemAccess, 93
- ValueMatrix, 270
 - addColumn, 280
 - addColumnScaledBy, 281
 - destroy, 281
 - getColumnCount, 282
 - getColumns, 283
 - getColumnsScaledBy, 284
 - getIndependentColumns, 284
 - getRowCount, 285
 - getScalingColumns, 285
 - getValue, 286
 - getValueMeaPoint, 287
 - getValueVector, 287
 - listColumns, 288
 - listColumnsScaledBy, 289
 - listIndependentColumns, 289
 - listScalingColumns, 290
 - removeValueMeaPoint, 290
 - removeValueVector, 291
 - setValue, 292
 - setValueMeaPoint, 293
 - setValueVector, 294
- ValueMatrix, 270
- ValueMatrix.addColumn, 280
- ValueMatrix.addColumnScaledBy, 281
- ValueMatrix.destroy, 282
- ValueMatrix.getColumnCount, 282
- ValueMatrix.getColumns, 283
- ValueMatrix.getColumnsScaledBy, 284
- ValueMatrix.getIndependentColumns, 284
- ValueMatrix.getRowCount, 285
- ValueMatrix.getScalingColumns, 285
- ValueMatrix.getValue, 286
- ValueMatrix.getValueMeaPoint, 287
- ValueMatrix.getValueVector, 287
- ValueMatrix.listColumns, 288
- ValueMatrix.listColumnsScaledBy, 289
- ValueMatrix.listIndependentColumns, 289
- ValueMatrix.listScalingColumns, 290
- ValueMatrix.removeValueMeaPoint, 290
- ValueMatrix.removeValueVector, 291
- ValueMatrix.setValue, 292
- ValueMatrix.setValueMeaPoint, 293
- ValueMatrix.setValueVector, 294
- WARNING, 335
- withUnit
 - ApplicationAttribute, 115
- withValueFlag
 - ApplicationAttribute, 116